Refine Search

Search Results -

Terms	Documents
"michael, ohran".in.	0

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search:

Refine Search

Recall Text Clear Interrupt

Search History

DATE: Monday, February 26, 2007 **Purge Queries** Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L44	"michael, ohran".in.	0	L44
L43	L41 and (polic$ with protocol or polic$ near protocol or polic$ adj protocol)	18	L43
L42	L41 not @py >1999	2	L42
L41	L40 and (I/O driver or on/off driver)	71	L41
L40	L39 and servers	94	L40
L39	L38 and (shared near stor$ near node or shared with stor$ with node or shared adj stor$ adj node)	111	L39
L38	L37 and (network or www or internet)	10472	L38
L37	(mirror$ near data or mirror$ with data or mirror$ adj data)	28742	L37
L36	370/1	739	L36
L35	711/1	1139	L35
L34	714/1	1147	L34

L33	709/1	580	L33
L32	709/232	3541	L32
L31	709/214	995	L31
L30	370/417	602	L30
L29	370.clas.	107378	L29
L28	714.clas.	58251	L28
L27	714/6	3484	L27
L26	714/4	3329	L26
L25	714.clas.	58251	L25
L24	711/147	2490	L24
L23	711/111	1804	L23
L22	711/114	4102	L22
L21	707/104.1	7829	L21
L20	707/10	14255	L20
L19	707/9	3452	L19
L18	707/8	2925	L18
L17	707/1	9142	L17
L16	707/200	5680	L16
L15	711.clas.	33552	L15
L14	707.clas.	41329	L14
L13	709.clas.	51976	L13
L12	709/200	4727	L12
L11	709/229	7225	L11
L10	709/227	7679	L10
L9	709/224	11196	L9
L8	709/212	933	L8
L7	709/217	10130	L7
L6	709/216	1321	L6
L5	709/201	5958	L5
L4	709/204	3886	L4
L3	709/203	13718	L3
L2	709/213	2804	L2
L1	709/219	9272	L1

END OF SEARCH HISTORY

Refine Search

Search Results -

Terms	Documents
7103797.pn.	4

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search:

Refine Search

Recall Text Clear Interrupt

Search History

DATE: Monday, February 26, 2007 **Purge Queries** Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L15	7103797.pn.	4	L15
DB=USPT; PLUR=YES; OP=OR			
L14	'5644784'.pn.	1	L14
L13	'5790539'.pn.	1	L13
L12	'5787084'.pn.	1	L12
L11	'5781549'.pn.	1	L11
L10	'5748631'.pn.	1	L10
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L9	6839349.pn.	2	L9
L8	731025.pn.	4	L8
L7	5771344.pn.	2	L7
L6	5799141.pn.	2	L6
L5	5555371.pn.	2	L5

DB=USPT; PLUR=YES; OP=OR

L4	("6324654")[URPN]	63	L4
L3	(5889935 | 6044444 | 5555371 | 6052797 | 5771344 | 5799141 | 5544347 | 5901327 | 5742792 | 5933653)![PN]	10	L3
L2	("6324654")[PN]	1	L2

DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR

L1	6324654.pn.	2	L1

END OF SEARCH HISTORY

Terms used **mirroring network data policing protocol** Found **3,018** of **198,146**

Sort results by relevance
Display results expanded form
Save results to a Binder
Search Tips
Open results in a new window
Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Policing congestion response in an internetwork using re-feedback

Bob Briscoe, Arnaud Jacquet, Carla Di Cairano-Gilfedder, Alessandro Salvatori, Andrea Soppera, Martin Koyabe
August 2005 **ACM SIGCOMM Computer Communication Review , Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications SIGCOMM '05**, Volume 35 Issue 4
Publisher: ACM Press
Full text available: pdf(361.70 KB) Additional Information: full citation, abstract, references, index terms

This paper introduces a novel feedback arrangement, termed re-feedback. It ensures metrics in data headers such as time to live and congestion notification will arrive at each relay carrying a truthful prediction of the remainder of their path. We propose mechanisms at the network edge that ensure the dominant selfish strategy of both network domains and end-points will be to set these headers honestly and to respond correctly to path congestion and delay, despite conflicting interests. Although ...

Keywords: QoS, characterisation, congestion, incentives, policing

2 Stateful distributed interposition

John Reumann, Kang G. Shin
February 2004 **ACM Transactions on Computer Systems (TOCS)**, Volume 22 Issue 1
Publisher: ACM Press
Full text available: pdf(833.84 KB) Additional Information: full citation, abstract, references, index terms

Interposition-based system enhancements for multitiered servers are difficult to build because important system context is typically lost at application and machine boundaries. For example, resource quotas and user identities do not propagate easily between cooperating services that execute on different hosts or that communicate with each other via intermediary services. Application-transparent system enhancement is difficult to achieve when such context information is obscured by complex servic ...

Keywords: Distributed computing, component services, distributed context, multitiered services, operating systems, server consolidation

3 Accelerating telnet performance in wireless networks

Barron Housel, Ian Shields

August 1999 **Proceedings of the 1st ACM international workshop on Data engineering for wireless and mobile access MobiDe '99**
Publisher: ACM Press
Full text available: pdf(933.49 KB) Additional Information: full citation, references, index terms

Keywords: caching, compression, data reduction, emulation, mobile, sessions, telnet, wireless

4 Axon: network virtual storage design
James P. G. Sterbenz, Gurudatta M. Parulkar
April 1990 **ACM SIGCOMM Computer Communication Review**, Volume 20 Issue 2
Publisher: ACM Press
Full text available: pdf(1.16 MB) Additional Information: full citation, abstract, citings, index terms

This paper describes the design of *network virtual storage* (NVS) in the Axon host communication architecture for distributed applications. The Axon project is investigating an integrated design of host architecture, operating systems, and communication protocols to allow applications to utilise the high bandwidth provided by the next generation of communication networks. NVS extends segmented paged virtual storage management and address translation mechanisms to include segments ...

5 Database session 4: heterogeneous and distributed systems: A reliable storage management layer for distributed information retrieval systems
Charles L. A. Clarke, Philip L. Tilker, Allen Quoc-Luan Tran, Kevin Harris, Antonio S. Cheng
November 2003 **Proceedings of the twelfth international conference on Information and knowledge management CIKM '03**
Publisher: ACM Press
Full text available: pdf(165.86 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a storage management layer that facilitates the implementation of parallel information retrieval systems, and related applications, on networks of workstations. The storage management layer automates the process of adding and removing nodes, and implements a dispersed mirroring strategy to improve reliability. When nodes are added and removed, the document collection managed by the system is redistributed for load balancing purposes. The use of dispersed mirroring minimizes the impact ...

Keywords: cluster computing, distributed information retrieval, self-managing systems

6 Memory optimization: Automatic data partitioning for the agere payload plus network processor
Steve Carr, Philip Sweany
September 2004 **Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems CASES '04**
Publisher: ACM Press
Full text available: pdf(195.89 KB) Additional Information: full citation, abstract, references, index terms

With the ever-increasing pervasiveness of the Internet and its stringent performance requirements, network system designers have begun utilizing specialized chips to increase the performance of network functions. To increase performance, many more advanced functions, such as traffic shaping and policing, are being implemented at the network interface layer to reduce delays that occur when these functions are handled by a general-purpose CPU. While some designs use ASICs to handle network functio ...

Keywords: network processors, partitioning, scheduling

7 Serverless network file systems

Thomas E. Anderson, Michael D. Dahlin, Jeanna M. Neefe, David A. Patterson, Drew S. Roselli, Randolph Y. Wang
February 1996 **ACM Transactions on Computer Systems (TOCS)**, Volume 14 Issue 1
Publisher: ACM Press

Full text available: pdf(2.69 MB) Additional Information: full citation, abstract, references, citings, index terms

We propose a new paradigm for network file system design: serverless network file systems. While traditional network file systems rely on a central server machine, a serverless system utilizes workstations cooperating as peers to provide all file system services. Any machine in the system can store, cache, or control any block of data. Our approach uses this location independence, in combination with fast local area networks, to provide better performance and scalability th ...

Keywords: RAID, log cleaning, log structured, log-based striping, logging, redundant data storage, scalable performance

8 Illustrative risks to the public in the use of computer systems and related technology

Peter G. Neumann
January 1996 **ACM SIGSOFT Software Engineering Notes**, Volume 21 Issue 1
Publisher: ACM Press
Full text available: pdf(2.54 MB) Additional Information: full citation

9 Illustrative risks to the public in the use of computer systems and related technology

Peter G. Neumann
January 1994 **ACM SIGSOFT Software Engineering Notes**, Volume 19 Issue 1
Publisher: ACM Press
Full text available: pdf(2.24 MB) Additional Information: full citation, citings, index terms

10 Performance: A QoS service for IP video applications on demand over DTM

Cláudia J. Barenco, Arturo Azcorra Saloña, José Ignacio Moreno
April 2001 **ACM SIGCOMM Computer Communication Review**, Volume 31 Issue 2 supplement
Publisher: ACM Press
Full text available: pdf(2.26 MB) Additional Information: full citation, abstract, references

The Differentiated Services model (DiffServ) provides a great flexibility in defining a variety of services through PHBs (*Per Hop Behaviors*) and traffic conditioners. It fits in well with the Integrated Services model (IntServ), jointly offering features such as: QoS signaling; admission control; channel management; assignment of resources (buffer and bandwidth); sorter configuration; and establishment of traffic agreements. With this integration you can have a scalable, flexible, and dyn ...

Keywords: DTM, DiffServ, IntServ, QoS, video on demand

11 Columns: Risks to the public in computers and related systems

Peter G. Neumann
January 2001 **ACM SIGSOFT Software Engineering Notes**, Volume 26 Issue 1
Publisher: ACM Press
Full text available: pdf(3.24 MB) Additional Information: full citation

12 Designing families of data types using exemplars
Wilf R. LaLonde
April 1989 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 11 Issue 2
Publisher: ACM Press
Full text available: pdf(2.90 MB) Additional Information: full citation, abstract, references, citings, index terms

Designing data types in isolation is fundamentally different from designing them for integration into communities of data types, especially when inheritance is a fundamental issue. Moreover, we can distinguish between the design of families—integrated types that are variations of each other—and more general communities where totally different but cohesive collections of types support specific applications (e.g., a compiler). We are concerned with the design of integrated familie ...

13 Rethinking the design of the Internet: the end-to-end arguments vs. the brave new world
Marjory S. Blumenthal, David D. Clark
August 2001 **ACM Transactions on Internet Technology (TOIT)**, Volume 1 Issue 1
Publisher: ACM Press
Full text available: pdf(176.33 KB) Additional Information: full citation, abstract, references, citings, index terms

This article looks at the Internet and the changing set of requirements for the Internet as it becomes more commercial, more oriented toward the consumer, and used for a wider set of purposes. We discuss a set of principles that have guided the design of the Internet, called the end-to-end arguments, and we conclude that there is a risk that the range of new requirements now emerging could have the consequence of compromising the Internet's original design principles. Were ...

Keywords: ISP, Internet, end-to-end argument

14 Tussle in cyberspace: defining tomorrow's internet
David D. Clark, John Wroclawski, Karen R. Sollins, Robert Braden
June 2005 **IEEE/ACM Transactions on Networking (TON)**, Volume 13 Issue 3
Publisher: IEEE Press
Full text available: pdf(414.87 KB) Additional Information: full citation, abstract, references, index terms

The architecture of the Internet is based on a number of principles, including the self-describing datagram packet, the end-to-end arguments, diversity in technology and global addressing. As the Internet has moved from a research curiosity to a recognized component of mainstream society, new requirements have emerged that suggest new design principles, and perhaps suggest that we revisit some old ones. This paper explores one important reality that surrounds the Internet today: different stakeh ...

Keywords: competition, design principles, economics, network architecture, trust, tussle

15 The platform for privacy preference as a social protocol: An examination within the

U.S. policy context
Harry Hochheiser
November 2002 **ACM Transactions on Internet Technology (TOIT)**, Volume 2 Issue 4
Publisher: ACM Press
Full text available: pdf(241.03 KB) Additional Information: full citation, abstract, references, citings, index terms

As a "social protocol" aimed at providing a technological means to address concerns over Internet privacy, the Platform for Privacy Preferences (P3P) has been controversial since its announcement in 1997. In the U.S., critics have decried P3P as an industry attempt to avoid meaningful privacy legislation, while developers have portrayed the proposal as a tool for helping users make informed decisions about the impact of their Web surfing choices. This dispute touches upon the privacy model under ...

Keywords: P3P, Privacy, social protocols

16 Papers: An ECN probe-based connection acceptance control
Tom Kelly
July 2001 **ACM SIGCOMM Computer Communication Review**, Volume 31 Issue 3
Publisher: ACM Press
Full text available: pdf(1.20 MB) Additional Information: full citation, abstract, references

Connection acceptance control is a mechanism which can be used to moderate the load placed on a network by turning away connection requests during times of overload. Traditionally these mechanisms have been implemented by setting up state within a network using signalling protocols, such as the IETF's Resource Reservation Protocol. There have been recent proposals for admission control based on end-systems probing the network to infer network load. These distributed algorithms often have less ro ...

17 Papers: TCP congestion control with a misbehaving receiver
Stefan Savage, Neal Cardwell, David Wetherall, Tom Anderson
October 1999 **ACM SIGCOMM Computer Communication Review**, Volume 29 Issue 5
Publisher: ACM Press
Full text available: pdf(783.47 KB) Additional Information: full citation, abstract, references, citings

In this paper, we explore the operation of TCP congestion control when the receiver can misbehave, as might occur with a greedy Web client. We first demonstrate that there are simple attacks that allow a misbehaving receiver to drive a standard TCP sender arbitrarily fast, without losing end-to-end reliability. These attacks are widely applicable because they stem from the sender behavior specified in RFC 2581 rather than implementation bugs. We then show that it is possible to modify TCP to eli ...

18 Illustrative risks to the public in the use of computer systems and related technology
Peter G. Neumann
January 1992 **ACM SIGSOFT Software Engineering Notes**, Volume 17 Issue 1
Publisher: ACM Press
Full text available: pdf(1.65 MB) Additional Information: full citation, citings, index terms

19 An Approach to Protocol Modeling and Validation
F. Cicirelli, A. Furfaro, L. Nigro
April 2006 **Proceedings of the 39th annual Symposium on Simulation - Volume 00 ANSS '06**
Publisher: IEEE Computer Society

Full text available: pdf(348.31 KB) Additional Information: full citation, abstract

This paper describes an approach to modeling and analysis of complex time-dependent systems specified by modular Time Petri Nets (TPNs). The approach is supported by a Java tool TPN Designer- which permits visual modeling, debugging and discrete-event simulation. The tool is characterized by its modularity and hierarchical constructs, a scripting language for controlling model configuration and scalability, and a monitoring and watching sub system for collecting simulation output. TPN Designer p ...

20 Social Analyses of Computing: Theoretical Perspectives in Recent Empirical Research

Rob Kling

March 1980 **ACM Computing Surveys (CSUR)**, Volume 12 Issue 1

Publisher: ACM Press

Full text available: pdf(3.98 MB) Additional Information: full citation, references, citings, index terms

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Terms used **mirroring network data** Found **3,018** of **198,146**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Query optimization in distributed networks of autonomous database systems
Fragkiskos Pentaris, Yannis Ioannidis
June 2006 **ACM Transactions on Database Systems (TODS)**, Volume 31 Issue 2
Publisher: ACM Press
Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, index terms

Large-scale distributed environments, where each node is completely autonomous and offers services to its peers through external communication, pose significant challenges to query processing and optimization. Autonomy is the main source of the problem, as it results in lack of knowledge about any particular node with respect to the information it can produce and its characteristics, for example, cost of production or quality of produced results. In this article, inspired by e-commerce technolog ...

Keywords: Query optimization

2 Sensor networks II: Double rulings for information brokerage in sensor networks
Rik Sarkar, Xianjin Zhu, Jie Gao
September 2006 **Proceedings of the 12th annual international conference on Mobile computing and networking MobiCom '06**
Publisher: ACM Press
Full text available: pdf(891.39 KB) Additional Information: full citation, abstract, references, index terms

We study the problem of *information brokerage* in sensor networks, where information consumers (sinks,users)search for data acquired by information producers (sources). In-network storage such as geographical hash table (GHTs) has been proposed to store data at rendezvous nodes for consumers to retrieve. In this paper, we propose a *double rulings* scheme which stores data replica at a curve instead of one or multiple isolated sensors. The consumer travels along another curve which gu ...

Keywords: data-centric routing, double rulings, information storage and retrieval, sensor networks

3 SIGGRAPH 2005 emerging technologies projects: Mirror_SPACE project: system of real time events and installation
Brigitta Zics
July 2005 **ACM SIGGRAPH 2005 Emerging technologies SIGGRAPH '05**

Publisher: ACM Press
Full text available: pdf(198.65 KB) Additional Information: full citation, abstract

Interactive networked installation, which projects a personal, virtual mirror-image onto the screen, with the aid of the combination of the face of the visitor and data collected simultaneously from the Internet. This image behaves like the physical presence of a real mirror-image: it changes its position, dimensions and features according to the movement of the viewer. The common mirror-representations of individual visitors also interact with each other, and their audiovisual representation is ...

4 An object-based infrastructure for program monitoring and steering
Greg Eisenhauer, Karsten Schwan
August 1998 **Proceedings of the SIGMETRICS symposium on Parallel and distributed tools SPDT '98**
Publisher: ACM Press
Full text available: pdf(1.50 MB) Additional Information: full citation, references, citings, index terms

5 A High Availability Clustering Solution
Phil Lewis
August 1999 **Linux Journal**
Publisher: Specialized Systems Consultants, Inc.
Full text available: html(34.77 KB) Additional Information: full citation, abstract, index terms

Mr. Lewis tells us how he designed and implemented a simple high-availability solution for his company

6 Storage systems: Lessons and challenges in automating data dependability
Kimberly Keeton, Dirk Beyer, Jeffrey Chase, Arif Merchant, Cipriano Santos, John Wilkes
September 2004 **Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC EW11**
Publisher: ACM Press
Full text available: pdf(175.67 KB) Additional Information: full citation, abstract, references, citings

Designing and managing dependable systems is a difficult endeavor. In this paper, we describe challenges in this vast problem space, including provisioning and allocating shared resources, adaptively managing system dependability, expressing dependability goals, interactively exploring the design space, and designing end-to-end service dependability. We outline the optimization-based approach we've used to tackle the data dependability portion of this space, and describe how we can extend that a ...

7 Time- and power-sensitive techniques: Video-streaming for fast moving users in 3G mobile networks
Anna Kyriakidou, Nikos Karelos, Alex Delis
June 2005 **Proceedings of the 4th ACM international workshop on Data engineering for wireless and mobile access MobiDE '05**
Publisher: ACM Press
Full text available: pdf(306.97 KB) Additional Information: full citation, abstract, references, index terms, review

The emergence of third-generation (3G) mobile networks offers new opportunities for the effective delivery of data with rich content including multimedia messaging and video-streaming. Provided that streaming services have proved highly successful over stationary networks in the past, we anticipate that the same trend will soon take place in 3G networks. Although mobile operators currently make available pertinent services, the available resources of the underlying networks for the delivery of r ...

Keywords: mobile multimedia services, rate adaptation, real-time streaming, streaming for moving users

8 Building and supporting a massive data infrastructure for the masses

Anurag Shankar, Gustav Meglicki, Jeff Russ, Haichuan Yang, E. Chris Garrison
November 2002 **Proceedings of the 30th annual ACM SIGUCCS conference on User services SIGUCCS '02**
Publisher: ACM Press

Full text available: pdf(526.33 KB) Additional Information: full citation, abstract, references, citings, index terms

On a typical university campus, the words "massive data storage" (MDS) usually bring to mind technology high-end, high performance computing (HPC) users might use. This is because academic supercomputer sites have traditionally provided a tightly interwoven HPC and high performance, MDS fabric to their users for decades. However, a new paradigm in data storage is now emerging where large, central, hierarchical storage management (HSM) services may play an increasingly important role in the non-H ...

Keywords: DCE, DFS, HPSS, HSM, distributed storage, hierarchical storage management, massive data storage, support

9 Using Handheld Devices in Synchronous Collaborative Scenarios

Jörg Roth, Claus Unger
January 2001 **Personal and Ubiquitous Computing**, Volume 5 Issue 4
Publisher: Springer-Verlag
Full text available: pdf(284.67 KB) Additional Information: full citation, abstract, index terms

In this paper we present a platform specially designed for groupware applications running on handheld devices. Common groupware platforms request desktop computers as underlying hardware platforms. The fundamentally different nature of handheld devices has a great impact on the platform, e.g. resource limitations have to be considered, the network is slow and unstable. Often, personal data are stored on handheld devices, thus mechanisms have to ensure privacy. These considerations led to the Qui ...

10 DNS: On the responsiveness of DNS-based network control

Jeffrey Pang, Aditya Akella, Anees Shaikh, Balachander Krishnamurthy, Srinivasan Seshan
October 2004 **Proceedings of the 4th ACM SIGCOMM conference on Internet measurement IMC '04**
Publisher: ACM Press

Full text available: pdf(255.96 KB) Additional Information: full citation, abstract, references, citings, index terms

For the last few years, large Web content providers interested in improving their scalability and availability have increasingly turned to three techniques: mirroring, content distribution, and ISP multihoming. The Domain Name System (DNS) has gained a prominent role in the way each of these techniques directs client requests to achieve the goals of scalability and availability. The DNS is thought to offer the transparent and agile control necessary to react quickly to ISP link failures or ph ...

Keywords: DNS, network control, time-to-live

11 Joint session: Building scalable and robust peer-to-peer overlay networks for broadcasting using network coding

Kamal Jain, László Lovász, Philip A. Chou
July 2005 **Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing PODC '05**
Publisher: ACM Press
Full text available: pdf(526.03 KB) Additional Information: full citation, abstract, references, index terms

We propose a scheme for building peer-to-peer overlay networks for broadcasting using network coding. The scheme addresses many practical issues such as scalability, robustness, constraints on bandwidth, and locality of decisions. We analyze the system theoretically and prove near optimal bounds on the parameters defining robustness and scalability. As a result we show that the effects of failures are contained locally, allowing the network to grow exponentially with server load. We also argue t ...

Keywords: Azuma, coding, file distribution, inequalities, multicast, network, overlay, peer-to-peer, security

12 Subverting Structure: Data-Driven Diagram Generation
Gene Golovchinsky, Klaus Reichenberger, Thomas Kamps
October 1995 **Proceedings of the 6th conference on Visualization '95 VIS '95**
Publisher: IEEE Computer Society
Full text available: pdf(848.62 KB) Publisher Site Additional Information: full citation, abstract, citings

Diagrams are data representations that convey information predominantly through combinations of graphical elements rather than through other channels such as text or interaction. We have implemented a prototype called AVE (Automatic Visualization Environment) that generates diagrams automatically based on a generative theory of diagram design. According to this theory, diagrams are constructed based on the data to be visualized rather than by selection from a predefined set of diagrams. This app ...

13 IDMaps: a global internet host distance estimation service
Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, Lixia Zhang
October 2001 **IEEE/ACM Transactions on Networking (TON)**, Volume 9 Issue 5
Publisher: IEEE Press
Full text available: pdf(267.64 KB) Additional Information: full citation, abstract, references, citings, index terms

There is an increasing need to quickly and efficiently learn network distances, in terms of metrics such as latency or bandwidth, between Internet hosts. For example, Internet content providers often place data and server mirrors throughout the Internet to improve access latency for clients, and it is necessary to direct clients to the nearest mirrors based on some distance metric in order to realize the benefit of mirrors. We suggest a scalable Internet-wide architecture, called IDMaps, which m ...

Keywords: Distributed algorithms, modeling, network service, scalability

14 Specification and verification of network managers for large internets
D. L. Cohrs, B. P. Miller
August 1989 **ACM SIGCOMM Computer Communication Review , Symposium proceedings on Communications architectures & protocols SIGCOMM '89**, Volume 19 Issue 4
Publisher: ACM Press
Full text available: pdf(1.56 MB) Additional Information: full citation, abstract, references, citings, index terms

Large internet environments are increasing the difficulty of network management. Integrating increasing numbers of autonomous subnetworks (each with an increasing number of hosts) makes it more difficult to determine if the network managers of the subnetworks will interoperate correctly. We propose a high level, formal specification language, NMSL, as an aid in solving this problem. NMSL has two aspects of operation, a descriptive aspect and a prescriptive aspect. In its descriptive aspect, ...

15 A case for network musical performance

John Lazzaro, John Wawrzynek

January 2001 **Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video NOSSDAV '01**

Publisher: ACM Press

Full text available: pdf(214.14 KB) Additional Information: full citation, abstract, references, citings, index terms

A Network Musical Performance (NMP) occurs when a group of musicians, located at different physical locations, interact over a network to perform as they would if located in the same room. In this paper, we present a case for NMP as a practical Internet application, and describe a method to ameliorate the effect of late and lost packets on NMP. We describe an NMP system that embodies this concept, that combines several existing standards (MIDI, MPEG 4 Structured Audio, RTP/AVP, and SIP) wi ...

16 Experimental testbeds and data: The changing usage of a mature campus-wide wireless network

Tristan Henderson, David Kotz, Ilya Abyzov

September 2004 **Proceedings of the 10th annual international conference on Mobile computing and networking MobiCom '04**

Publisher: ACM Press

Full text available: pdf(625.48 KB) Additional Information: full citation, abstract, references, citings, index terms

Wireless Local Area Networks (WLANs) are now commonplace on many academic and corporate campuses. As "Wi-Fi" technology becomes ubiquitous, it is increasingly important to understand trends in the usage of these networks.This paper analyzes an extensive network trace from a mature 802.11 WLAN, including more than 550 access points and 7000 users over seventeen weeks. We employ several measurement techniques, including syslogs, telephone records, SNMP polling and tcpdump packet sniffing. This is ...

Keywords: 802.11, VoIP, WLAN, Wi-Fi, telephony, voice, wireless network

17 Wireless sensor networks: Hood: a neighborhood abstraction for sensor networks

Kamin Whitehouse, Cory Sharp, Eric Brewer, David Culler

June 2004 **Proceedings of the 2nd international conference on Mobile systems, applications, and services MobiSys '04**

Publisher: ACM Press

Full text available: pdf(788.46 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper proposes a neighborhood programming abstraction for sensor networks, wherein a node can identify a subset of nodes around it by a variety of criteria and share state with those nodes. This abstraction allows developers to design distributed algorithms in terms of the neighborhood abstraction itself, instead of decomposing them into component parts such as messaging protocols, data caches, and neighbor lists. In those applications that are already neighborhood-based, this abstraction i ...

Keywords: abstraction, data sharing, distributed algorithms, neighborhood, sensor

networks

18 Building objects and interactors for collaborative interactions with GASP

Thierry Duval, David Margery

September 2000 **Proceedings of the third international conference on Collaborative virtual environments CVE '00**

Publisher: ACM Press

Full text available: pdf(429.82 KB) Additional Information: full citation, references, index terms

Keywords: distributed interactions, distributed virtual reality, human-computer interfaces, synchronous cooperation

19 The MIRRORS/II simulator

C. Lynne D'Autrechy, James A. Reggia

June 1987 **Proceedings of the 20th annual symposium on Simulation ANSS '87**

Publisher: IEEE Computer Society Press

Full text available: pdf(606.90 KB) Additional Information: full citation, abstract, references, index terms

MIRRORS/II is a software system for developing connectionist models. This paper discusses the unique features of MIRRORS/II: application independence, specification language, multiple target languages, extensibility, and accumulative resources. A detailed example of the use of MIRRORS/II is given. Finally, MIRRORS/II is compared with other existing connectionist modelling systems and future plans are discussed.

20 Session 3: inference and statistical analysis: A signal analysis of network traffic anomalies

Paul Barford, Jeffery Kline, David Plonka, Amos Ron

November 2002 **Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment IMW '02**

Publisher: ACM Press

Full text available: pdf(1.52 MB) Additional Information: full citation, abstract, references, citings, index terms

Identifying anomalies rapidly and accurately is critical to the efficient operation of large computer networks. Accurately characterizing important classes of anomalies greatly facilitates their identification; however, the subtleties and complexities of anomalous traffic can easily confound this process. In this paper we report results of signal analysis of four classes of network traffic anomalies: outages, flash crowds, attacks and measurement failures. Data for this study consists of IP flow ...

Sign in

Google

Web Images Video News Maps more »

network data mirroring >1996 Search Advanced Search Preferences

Web

Results 1 - 10 of about 1,060,000 for network data mirroring >1996. (0.12 seconds)

Network Data Mirroring

Network data mirroring complements these traditional backup methods by ... **Network data mirroring** is a newer technology, complementary to tape backup and ...
www.drj.com/articles/Win98/welco.htm - 16k - Cached - Similar pages

5 Database mirroring

1996]). Under this scheme, the software used by the ADS mirrors is first compiled ... Figure 8: Schematic representation of **network mirroring** models used to ...
ads.harvard.edu/pubs/A+AS/2000A+AS..143...85A/node5.html - 24k - Cached - Similar pages

[PDF] On the Performance and Scalability of a Data Mirroring Approach ...

File Format: PDF/Adobe Acrobat - View as HTML
3.2 Experiment 2: **Network**. Throughput. **Data mirroring** under the model in ... [12] F. Leitner, "**Mirror** 1.0",**1996**. [13] A. Z. Spector and M. L. Kazar, ...
loci.cs.utk.edu/dsi/netstore99/docs/papers/dempsey.pdf - Similar pages

Computer network remote data mirroring system - Patent 6324654

A computer **network** remote **data mirroring** system writes update **data** both to a local **data** device and to a ... 5555371, Sep., **1996**, Duyanovich et al. 714/13. ...
www.freepatentsonline.com/6324654.html - 115k - Cached - Similar pages

Remote data mirroring system having a remote link adapter - Patent ...

08/052039 filed Apr. 23, 1993, entitled REMOTE **DATA MIRRORING** (U.S. Pat. No. 5544347 issued Aug. 6, **1996**), which are all incorporated herein by reference. ...
www.freepatentsonline.com/7073090.html - 104k - Cached - Similar pages

Sybase Ports Data Mirroring System to Oracle

The **mirroring** system enhances them by reducing **network** costs, accelerating recovery time, and guaranteeing **data** integrity." ...
www.eweek.com/article2/0,1895,1953351,00.asp - 101k - Cached - Similar pages

[PDF] World's First Transatlantic Implementation of True Data Mirroring

File Format: PDF/Adobe Acrobat - View as HTML

maximizing IP **network** investments. Background. In **1996**, international shipping companies ... completely reliable large scale **data mirroring** ...
www.inrange.com/.../mkt/sstory/ss_worlds_first_transatlantic_implementation_of_true_**data_mirroring**_787.pdf - Similar pages

HPCwire Article # 19001
"Delivering storage networking applications such as **data mirroring** over wide area ... P&O Nedlloyd Container Line Limited, was formed in December **1996**, ...
www.hpcwire.com/hpc-bin/artread.pl?direction=Current&articlenumber=19001 - 9k - Cached - Similar pages

Sys Admin > Real-Time Remote **Data Mirroring**
issue of The Perl Journal from **1996**-2002 in one convenient CD-ROM! ... Before I get into the **data-mirroring** techniques, I will explain the types of **data** ...
www.samag.com/documents/s=9364/sam0106sc/0106c.htm - 22k - Cached - Similar pages

Article On Sun Solaris Oracle Database **Mirroring** Software ...
Brocade and the File Area **Network** Solution Profile by Brocade ... through 2004 and every issue of The Perl Journal from **1996**-2002 in one convenient CD-ROM! ...
whitepaper.unixreview.com/.../Article%20On%20Sun%20Solaris%20Oracle%20**Data**base%20**Mirroring**%20Software - Similar pages

Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

United States Patent 6839349

Mirroring in a stacked network switch configuration

US Patent Issued on January 4, 2005

Inventor(s)

Mohan Kalkunte
Shekhar Ambe
Anders Johnson

Assignee

Broadcom Corporation

Application

No. 731025 filed on 2000-12-07

Current US Class

370/390 , 709/238

Field of Search

370/390 , 370/391 , 370/401 , 370/465 , 709/238

Examiners

Primary: Douglas Olms
Assistant: Robert W Wilson

Attorney, Agent or Firm

Squire, Sanders & Dempsey L.L.P.

Abstract

A method of mirroring data to a mirrored to port in a plurality of switches. The method has the steps of determining if data was sent to all of said plurality of switches; determining if said data was sent to a mirrored to port (MTP); and resending said data to all of said plurality of switches if mirroring is enabled and said data was not sent to said MTP.

US Patent References

5390173
5414704
5423015
5459717
5473607
5499295
5524254

Other References

"Queue Management for Shared Buffer and Shared Multi-buffer ATM Switches," Yu-Sheng Lin et al., Department of Electronics Engineering & Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, R.O.C., Mar. 24, 1996, pp. 688-695.

5555398
5568477
5579301
5644784
5652579
5696899
5742613
5748631
5781549
5787084
5790539
5802052
5802287
5825772
5828653
5831980
5842038
5845081
5887187
5892922
5898687
5909686
5918074
5940596
5987507
6011795
6041053
6061351
6119196
6175902
6185185
6425015
6707817
6707818

"Computer Networks," A.S. Tanenbaum, Prentice-Hall Int., USA, XP-002147300(1998), Sec. 5.2-Sec. 5.3, pp. 309-320.

"A High-Speed CMOS Circuit for 1.2-Gb/s 16×16 ATM Switching," Alain Chemarin et al. 8107 IEEE Journal of Solid-State Circuits 27(1992)Jul., No. 7, New York, US, pp. 1116-1120.

"Local Area Network Switch Frame Lookup Technique for Increased Speed and Flexibility," 700 IBM Technical Disclosure Bulletin 38(1995)Jul., No. 7, Armonk, NY, US, pp. 221-222.

"A 622-Mb/s 8×8 ATM Switch Chip Set with Shared Multibuffer Architecture," Harufusa Kondoh et al., 8107 IEEE Journal of Solid-State Circuit 28(1993)Jul., No. 7, New York, US, pp. 808-814.

"Catalyst 8500 CSR Architecture." White Paper XP-002151999, Cisco Systems Inc. 1998, pp. 1-19.

Foreign Patent References

4-189023 JP. Jul., 1992
0312917 EP. Apr., 1989
0465090 EP. Jan., 1992
0752796 EP. Jan., 1997
0849917 EP. Jun., 1998
0853441 EP. Jul., 1998
0854606 EP. Jul., 1998
0859492 EP. Aug., 1998
0862349 EP. Sep., 1998
0907300 EP. Apr., 1999
2 725 573 FR. Apr., 1996
WO 98/09473 WO Mar., 1998
WO 99/00938 WO Jan., 1999
WO 99/00939 WO Jan., 1999
WO 99/00944 WO Jan., 1999
WO 99/00945 WO Jan., 1999

WO 99/00948	WO	Jan., 1999
WO 99/00949	WO	Jan., 1999
WO 99/00950	WO	Jan., 1999
WO9900936	WO.	Jun., 2001
WO 99/09713	WO	Feb., 1999
WO 00/72533	WO	Nov., 2000
GB 2 333 429	GB	Jul., 1999

Home | Browse by Inventor | Browse by Date | Resources | Contact Us

United States Patent 7103797

Resource allocation throttling in remote data mirroring system

US Patent Issued on September 5, 2006

Inventor(s)
Steven B. Wahl
Michael W. Losh

Assignee
EMC Corporation

Application
No. 10613114 filed on 2003-07-03

Current US Class
714/6 , 711/113

Examiners
Primary: Robert Beausoliel
Assistant: Michael Maskulinski

Attorney, Agent or Firm
Van Pelt, Yi & James LLP

US Patent References
5371878
5379418
5511190
5544347
5555371
5742792
5742933
5771344
5799141
5889935
5901327
5903898
5909548

ABSTRACT CLAIMS DESCRIPTION FULL TEXT

Abstract

A computer network remote data mirroring system writes update data both to a local data device and to a local, chronologically sequenced journal storage area, or writelog device. A graphical user interface enables a user to create and configure throttles, which are user-defined tests and actions evaluated by the primary mirror daemon to regulate network bandwidth, CPU, and writelog device utilization during data update mirroring. Network bandwidth throttling enables a predetermined portion of the network bandwidth to be assigned to remote data mirroring based on user-selected criteria. CPU throttling enables a user to control the amount of time the local data storage unit will wait prior to returning control to applications after an update. Writelog device throttling prevents a memory overflow condition by

5933653 6018778 6044444 6052797 6105029	dynamically assigning memory to the writelog device by chaining writelog device extensions to the writelog device.

Title: **Computer network remote data mirroring s**

Document Type and Number: United States Patent 6324654

Link to this Page: http://www.freepatentsonline.com/6324654.html

Abstract: A computer network remote data mirroring system writes update data l to a local, chronologically sequenced journal storage area, or writelog d system crashes, upon recovery or re-boot of the local computer system in the writelog device are written to the local data device to assure that data device is current. Additional memory or disk space is dynamically to prevent a memory overflow condition. The computer network remote structured to provide logical groups of local data device/writelog device on a local computer system monitors the writelog device for data updat network in the same order in which it is stored to a receiving remote m computer system, which in turn commits the data updates to a mirror c primary and remote mirror daemons is initiated automatically for certai the basic operability of the overall computer system. A graphical user ir configure the logical groups and create throttles, as well as to monitor mirroring system. Network bandwidth throttling enables a predetermine bandwidth to be assigned to remote data mirroring based on user-selec

Inventors:	Wahl, Steven B.; Losh, Michael W.;
Application Number:	050676
Filing Date:	1998-03-30
Publication Date:	2001-11-27
View Patent Images:	**Login** or **Create Account (Free!)**
Related Patents:	**View patents that cite this patent**

Export Citation:	**Click for automatic bibliography generation**
Assignee:	Legato Systems, Inc. (Mountain View, CA)
Primary Class:	**714/6**
Other Classes:	707/204, 711/162, 714/15
International Classes:	G06F 012/00; G06F 017/30; G06F 012/16
Field of Search:	714/1,5,6,7,15,16,20 711/112,113,114,161,162 709/217,218 710/52,!

US Patent References:

5544347	Aug., 1996	Yanai et al.	714/6.
5555371	Sep., 1996	Duyanovich et al.	714/13.
5742792	Apr., 1998	Yanai et al.	711/162.
5771344	Jun., 1998	Chan et al.	714/7.
5799141	Aug., 1998	Galipeau et al.	714/13.
5889935	Mar., 1999	Ofek et al.	714/6.
5901327	May., 1999	Ofek	710/5.
5933653	Aug., 1999	Ofek	710/6.
6044444	Mar., 2000	Ofek	711/162.
6052797	Apr., 2000	Ofek et al.	714/6.

Other References:	Louderback, Jim, "Trailblazing the client/server jungle, part3: your har client/server performance", Jan. 1992, Data Based Advisor, v10, p. 80·
Primary Examiner:	Wright; Norman M.
Assistant Examiner:	Revak; Christopher
Attorney, Agent or Firm:	Workman, Nydegger & Seeley

Claims:

What is claimed is:

1. A computer network remote data mirroring apparatus comprising:

a primary computer system for executing an application which generates data, the prim comprising:

a local data device;

a writelog device including a cache memory and a dirty bit map disk drive device to wh cache memory to avoid a memory overflow condition, the local data device and the writ as a local data storage unit;

a device driver for controlling transfer of the data to and from the local data device and

a primary mirror daemon executed by the primary computer system, the primary mirro from the writelog device through the device driver;

a secondary computer system for replicating the data, the secondary computer system

a mirror device onto which the data is replicated; and

a remote mirror daemon executed by the secondary computer system; and

a network for interconnecting the primary computer system and the secondary comput and secondary mirror daemons communicate over the network so that the primary mir from the writelog device over the network to the remote mirror daemon and the remot data on the mirror device.

2. The apparatus as defined in claim 1 wherein the primary computer system further cc controller cards and at least two I/O busses and wherein the local data device and the I/O busses, whereby performance is improved due to reduced I/O bus contention durin the local data device and the writelog device.

3. The apparatus as defined in claim 1 wherein the primary computer system further cc devices and a plurality of writelog devices, each of the plurality of local data devices op plurality of writelog devices as one of a plurality of local data storage units.

4. The apparatus as defined in claim 3 wherein the plurality of local data storage units a logical groups and wherein each of the plurality of logical groups corresponds to one of said one of the plurality of mirror devices operating at a secondary computer system.

5. The apparatus as defined in claim 1 wherein the network is one of a local area netwc

6. The apparatus as defined in claim 5 wherein the network supports a conventional TC

7. The apparatus as defined in claim 1, wherein the device driver further operates to co local data device and the writelog device by:

writing updates to the local data device, thereby modifying the data stored in the local

substantially simultaneously to writing the updates, writing entries representing the up

8. In a primary computer system having a local data device to which data is written an writelog in which information representing updates to the data can be stored, the prima communication with a secondary computer system, a method for mirroring the data stc mirror device associated with the secondary computer system, comprising the steps of:

storing data in the local data device, with a mirrored copy of the data being stored in th the secondary computer system;

repeatedly conducting the steps of:

writing an update to the local data device, thereby modifying the data stored in the loc

substantially simultaneously with the step of writing the update, writing an entry repre writelog;

transmitting entries representing updates from the writelog to the secondary computer secondary computer system can write the updates represented by the entries to the mi the mirror device to thereby update the mirrored copy, the entries being transmitted to chronological order in which the entries were stored at the writelog;

experiencing a state in which writing more entries to the non-volatile writelog would cr

after experiencing said state, and substantially simultaneously with writing other updat writing other entries representing the other updates to one or more writelog pool devic primary computer and that provide additional data storage for the writelog.

9. A method as recited in claim 8, wherein:

the step of writing an update to the local data device is performed by transmitting the u using a first I/O bus; and

the step of writing an entry representing the update to the writelog is performed by tra writelog using a second I/O bus.

10. A method as recited in claim 8, further comprising the steps of:

receiving confirmation that a particular entry has been committed to the mirrored copy

allowing a new entry to overwrite the particular entry in the writelog based on said con

11. A method as recited in claim 8, further comprising the steps of:

experiencing a failure at the primary computer system, such that the data stored in the accessible; and

providing access to the mirrored copy of the data stored in the mirror device.

12. A computer network remote data mirroring apparatus comprising:

a primary computer system for executing an application which generates data, the prim comprising:

a local data device;

a writelog device that operates together with the local data device as a local data stora

one or more optional writelog pool devices that are chained into the primary computer additional data storage is needed for the writelog device;

a device driver for controlling transfer of the data to and from the local data device and

a primary mirror daemon executed by the primary computer system, the primary mirro from the writelog device through the device driver;

a secondary computer system for replicating the data, the secondary computer system

a mirror device onto which the data is replicated; and

a remote mirror daemon executed by the secondary computer system; and

a network for interconnecting the primary computer system and the secondary comput and secondary mirror daemons communicate over the network so that the primary mir from the writelog device over the network to the remote mirror daemon and the remot data on the mirror device.

13. The apparatus as defined in claim 12, wherein the writelog device comprises a disk

14. The apparatus as defined in claim 12, wherein:

the primary computer system further comprises at least two disk controller cards and a

the local data device and the writelog device are on separate I/O busses, whereby perf reduced I/O bus contention during parallel writes of the data to the local data device ar

15. The apparatus as defined in claim 12, wherein the primary computer system furthe data devices and a plurality of writelog devices, each of the plurality of local data devic of the plurality of writelog devices as one of a plurality of local data storage units.

16. The apparatus as defined in claim 15, wherein the plurality of local data storage uni of logical groups and wherein each of the plurality of logical groups corresponds to one said one of the plurality of mirror devices operating at a secondary computer system.

17. The apparatus as defined in claim 12, wherein the device driver further operates to

the local data device and the writelog device by:

writing updates to the local data device, thereby modifying the data stored in the local

substantially simultaneously to writing the updates, writing entries representing the up

---

Description:

BACKGROUND OF THE INVENTION

The present invention relates to computer hardware and software systems and, more p restoration of data for such a system in the event of a crash of the system or a disaste become inoperative for a period of time. When such a system crashes or becomes inop measures have been provided to recover or restore data. Specifically, the present inver apparatus which implement substantially real-time networked disk, or data, mirroring c and wide area networks (WANs) in a computer system, such as a SPARC Solaris 2.X en and other applications.

Various techniques are known for recovery or restoration of data in the event of a crash disaster which causes the computer system to become inoperative for an indefinite peri One technique that is known is to replicate data as the data is generated by an applicat the computer system. This technique is typically referred to as disk, or data, mirroring.

Heretofore, data mirroring has been achieved by one of several approaches. One appro mirroring utilizing redundant arrays of independent disks (RAID). Using the RAID appro execution of an application program is written to multiple storage devices, such as conv contemporaneously with storage of the data on a local input/output (I/O) data storage provide volume management software and a redundant storage device on which data is management software replicates data on the redundant storage device contemporaneo the local I/O data storage device. Both of these approaches typically provide synchrono characterized by miniscule delay in the replication of data for system recovery.

Considered in more detail, both RAID and volume management approaches typically pr asynchronous disk mirroring. In a synchronous disk mirroring architecture, such as pro management approach, disk updates are committed to each of the disk devices in the r to the application program. In the event that one of the disks goes out of service, the d the other disk devices in the mirror.

The RAID and volume management approaches can be implemented to protect data loc satisfactory for local disk mirroring for data recovery in the event of a local I/O disk fail they do not address the problem of catastrophic system failure or disaster which rende inoperative for an extended period of time or even permanently.

Another approach is to provide remote data mirroring in addition to local data mirrorinc data mirroring system is implemented both locally and remotely so that data generated application program is additionally communicated over a network to a remote location f data mirroring enables recovery of the local computer system in the event of a tempora transfer of data processing operations to a remote computer system if the local comput the outage is for a prolonged period of time, or a disaster permanently disables the loc data mirroring systems have been commercialized by companies such as International Equipment Corporation, and Data General Corporation in the past. Such remote data m one of several modes, including a synchronous mode, asynchronous mode, and near sy

Unfortunately, implementing synchronous data mirroring over a network raises serious than working with local data channels that can accept data at 5, 20, or 40 megabytes (data must travel over a much lower bandwidth channel, stretching out data transfer tin top of the much lower bandwidth, further slowing I/O turnaround times. Any practical e application program that has compared network file system (NFS) update performance readily illustrates this point. If networked disk mirroring is implemented using synchron performance is tremendously degraded.

On the other hand, implementing asynchronous disk mirroring over a network raises da

event of a disaster, the data on the remote, or secondary, computer system may be up what would be found on the local, or primary, computer system.

The near synchronous mode is a forced compromise between the synchronous and asyı synchronous data mirroring provides asynchronous remote data mirroring at a preset ir computer system to periodically halt execution of the application program at the preset the remote computer system is acknowledged.

Therefore, a remote data mirroring system which comprises an architecture configured needed. Furthermore, such a system is needed which addresses the problem of the limi communication for data over the network.

SUMMARY OF THE INVENTION

The present invention provides methods and apparatus for a novel synchronous, asyncl computer system remote disk, or data, mirroring system over a network. Fundamentall data mirroring system in accordance with the present invention comprises an architectı both to a local data device and to a local, chronologically sequenced journal storage are embodiment, the writelog device comprises a redundant data storage device, such as a embodiment, the writelog device comprises cache memory and a dirty bit map disk driv written from the cache memory to avoid a memory overflow condition. Advantageously data device and the writelog device is layered on top of the operating system environm network remote data mirroring system of the present invention ports to many commerc systems.

Once written to the local data device and the writelog device, I/O operation returns con delivers to the application I/O performance comparable to simple local disk mirroring.

A primary mirror daemon on the local, or primary, computer system monitors the write feeds the data over a network in the same order in which it is stored to a receiving rem or secondary, computer system, which in turn commits the data updates to a mirror de computer network remote data mirroring system in accordance with the present invent network configurations and is compatible with many different local and remote disk stoı

In accordance with the present invention, the writelog device is configured so that mor assigned to the writelog device to prevent a memory overflow condition which might ot the embodiment in which the writelog device comprises a disk drive device, additional c assigned or another disk storage device is chained into the local, or primary, computer which the writelog device comprises cache memory and a dirty bit map disk drive devic dynamically assigned or another disk drive is chained into the local, or primary, compul overflow.

Also in accordance with the present invention, the computer network remote data mirrc to provide volume grouping, or logical groups. Consequently, data at the local, or prima plurality of remote sites, as compared to known architectures which provide point-to-pc site) data mirroring. Accordingly, the computer network remote data mirroring system provides a master primary mirror daemon and associated child primary mirror daemons mirror daemon and associated remote mirror daemons, to process data for replication. preferably provided by the computer network remote data mirroring system in accordar for configuring the logical groups, as well as for monitoring performance of the remote

The computer network remote data mirroring system of the present invention additiona throttling. Bandwidth throttling enables a predetermined portion of the network bandwi data mirroring depending on the time of day or other criteria.

In accordance with the present invention, a method is provided for ensuring data integı while updates are occurring in parallel to two data storage devices, such as disk drive d disks are simultaneously updated through a device driver. Preferably, each disk is provi as a SCSI interface, to enhance reliability and speed of data updates. Consequently, da substantially simultaneously on both the local data device and in the writelog device. Tl the present invention accommodates any of three conditions that may arise in the even condition is that the same update data has been stored on both the local data device ar

second condition is that the update data was stored on the local data device, but failed device; and the third condition is that the update data was written to the writelog devic local data device. In accordance with the method of the present invention, the current writelog device, while the immediately preceding update is written to the local data dev system crashes, upon recovery or re-boot of the computer system, the two most currer are written to the local data device to assure that the data stored on the local data devi

Additionally, in accordance with the present invention, failure recovery with the primary initiated automatically for certain failures which do not affect the basic operability of the example, the computer network remote data mirroring system in accordance with the p automatically recovered upon power shutoff of one of the local, or primary, and remote systems or temporary loss of the network link.

Preferably, the computer network remote data mirroring system of the present inventic mode. Consequently, the primary and remote mirror daemons are able to take advanta bandwidth during data transfers, as one would find in a synchronous mode, yet this is p application. One drawback is that in the event of a disaster, the data on the secondary several seconds older than what would be found on the primary computer system. How application performance and data synchronicity presents the optimal compromise availa requirements. Alternatively, the computer network remote data mirroring system in acc invention can be operated in a synchronous mode to better assure data synchronicity a performance. The computer network remote data mirroring system of the present inver operated in a near synchronous mode to enable adjustment of the trade-off between ap synchronicity.

The computer network remote data mirroring system in accordance with the present in mission critical data. The computer network remote data mirroring system of the prese application performance by implementing asynchronous, synchronous, or near synchro network bandwidth throttling. It provides substantially real-time data mirroring over LA data offsite, yet does not impact application performance significantly. In the event of a data center out of service for hours or days, operations can be transferred to a seconda operating on an up-to-the-minute copy of the original data set.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other objectives and features and the concomitant advantages of the pr understood and appreciated by those skilled in the art in view of the description of the below in conjunction with the accompanying drawings. In the drawings:

FIG. 1 is a schematic diagram of the architecture of the components of one embodimen remote data mirroring system in accordance with the present invention;

FIG. 2 shows one embodiment of a writelog device configuration for incorporation into t

FIG. 3 shows the position of a device driver in the kernel of the system shown in FIG. 1 data storage unit in the system shown in FIG. 1;

FIG. 4 illustrates primary mirror daemon/remote mirror daemon protocol;

FIG. 5 is a schematic diagram of the architecture of logical groups;

FIGS. 6-9 illustrate various screens of a graphical user interface used to configure throt present invention;

FIG. 10 illustrates an example of network bandwidth throttling in accordance with the p

FIG. 11 illustrates chronological coherency through a comparison of writelog device ent from data disk technique; and

FIG. 12 is an example of a qdsperftool chart.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The following describes various embodiments of the computer network remote data mir with the present invention. The computer network remote data mirroring system of the coherent copy of application data on a primary computer system, that is, a copy that ca secondary computer system in a disaster recovery scenario. The computer network rem accordance with the present invention is designed to mirror disk-based data from devic devices on a secondary system, across any available TCP/IP network connection. Data i real-time to assure integrity between the two computer systems in the event of hardwa human intervention. The computer network remote data mirroring system of the preser result through time-sequenced transfers of data from the primary computer system to t over the network. Should a failure occur on the primary computer system, the seconda immediate access to contemporary application data. Both the primary and secondary co to have adequate amounts of disk storage and network bandwidth allocated to accomm to provide remote data mirroring.

The computer network remote data mirroring system in accordance with the present in computer hardware and software. In order to facilitate an understanding of the comput mirroring system of the present invention, the computer network remote data mirrorin exemplary implementation configured for a computer system platform utilizing a Solari The technical specifications for such a computer system include the following. The platf UltraSPARC system. The operating system is SPARC Solaris 2.5 or later. The required d software. There are no additional requirements for random access memory (RAM) for th is required for journal storage (the exact amount depends on specific requirements). An example, OpenWindows, CDE, or X11R6, must be installed to use qdsperftool, qdsconfi will be described in more detail later.

The software for the exemplary implementation is currently available as Qualix DataSta Group, Inc., located in San Mateo, Calif., to provide a network-based data mirroring sol SPARC environments. For example, the software can be provided on CD. The Solaris ins and pkginfo are used. The software is installed with the Solaris pkgadd command. The mirroring system in accordance with the present invention requires no modifications to binaries and is compatible with NFS, VxFS, VxVm, SDS, and other common file systems disk utilities that are not specific to a particular disk type. The exemplary implementatic DataStar, Network Mirroring Software, Version 1.4, Operations Guide, Solaris 2.X, Part 1997, available from Qualix Group, Inc., San Mateo, Calif., which is hereby incorporate reference.

Generally, the computer network remote data mirroring system of the present inventio numeral 10 in FIG. 1, comprises various hardware components. The various hardware c illustrated in FIG. 1 and described in detail below.

The computer network remote data mirroring system 10 comprises a local, or primary, primary computer system 12 provides primary application and data storage services to and normal operation, the primary computer system 12 runs applications and provides

The computer network remote data mirroring system 10 also comprises a remote, or se The secondary computer system 14 stores a copy of the data from the primary comput installation and normal operation, the secondary computer system 14 furnishes a mirro be more than one secondary computer system 14 in the configuration, each representir set, as will be described in more detail later.

Considered in more detail, the primary computer system 12 comprises a local data devi character special device (a disk partition or a managed volume) (but pertains to both th the block mode device) that provides storage for data on the primary computer system 16 comprises a disk drive device. Reads of disk data are always satisfied by direct acce This data is duplicated onto the secondary computer system 14.

The primary computer system 12 also comprises a writelog device 18 which is specified character devices (but pertains to both the special character device and the block mode computer system used for the journaling of time-sequenced writes to the local data dev writelog device 18 is preferably allocated for each local data device 16 so that a one-to between the local data devices and the writelog devices. The writelog device 18 mainta

data entries being sent across a network 20 to the secondary computer system 14, as later.

In one embodiment, the writelog device 18 comprises a redundant data storage device, this embodiment, the writelog device 18 is a disk-based journal for I/O data updates ar data device 16 managed by a device driver 22. In another embodiment, the writelog de memory and a dirty bit map disk drive device managed by the device driver 22, to whic cache memory to avoid a memory overflow condition In the embodiment in which cach requirements of the primary computer system 12 increase. In one exemplary implemer increase to 128 MB or more.

Peak data update rate is preferably used to determine writelog device sizing. In one ex minimum size of the writelog device 18 is 150 kilobytes (KB). However, the size of the at least two MB.

Referring to FIG. 2, the writelog device 18 is preferably organized as a time-sequenced data updates, or data entries, reside at the tail of the writelog device 18, and new data of the writelog device. When the head of the writelog device 18 grows to the end of the to the beginning of the device.

If the head of the writelog device 18 were to grow sufficiently beyond the tail that it wo writelog device overflow condition would result, a state that would invalidate the writel mirror." The computer network remote data mirroring system 10 prevents writelog dev shortly.

Each entry written to the writelog device 18 consists of data and a header. The header used by other system components, such as a timestamp, sequence number, device offs The oldest data entries are read from the end or tail of the writelog device 18 and sent new data entries are written to the beginning or head of the writelog device. A reservec writelog device 18, that contains metadata about the writelog device, is updated every written to or read from the writelog device or if a configurable period of time has elapse metadata. The writelog device 18 is a circular queue that allows new data entries to ov after the device driver 22 has received confirmation that the older data entry has been on the secondary computer system 14.

During operation, a writelog device 18 may be in danger of overflow if a primary mirror the primary computer system 12 is unable to allocate space for incoming data entries. device 18 becomes big enough that it would overwrite the tail, an overflow condition is overflow would "break the mirror" at the point in time of the overflow. Instead, the com mirroring system 10 avoids the overflow condition so that an overflow does not invalida device 18 or cause data corruption of any kind.

Specifically, in accordance with the present invention, the writelog device 18 is configur space is dynamically assigned to the writelog device to prevent a memory overflow con corrupt stored data. In the embodiment in which the writelog device 18 comprises a dis storage is dynamically assigned or another disk storage device is chained into the prim embodiment in which the writelog device 18 comprises cache memory and a dirty bit m disk storage is dynamically assigned or another disk storage device can be chained into 12 to prevent memory overflow.

Considered in more detail, the primary computer system 12 preferably comprises a writ which is a collection of writelog devices that are not in use and act as spares. These wri automatically chained into existing writelog devices which comprise the writelog device an overflow is imminent (i.e., an overflow condition occurs). In one exemplary impleme writelog devices can be placed in the writelog device extension pool 18A which compris

Preferably, as shown in FIG. 3, each local data device 16 and associated writelog device data storage unit 26. Each local data storage unit 26 is the means by which application access, or store data while executing the software that comprises the computer networ 10. This provides the mapping to and management of a specific local data device 16, th and the associated mirror device 32. The local data storage unit 26 is only defined on th Preferably, there is a local data storage unit 26 instance for each local data device/write

primary computer system 12. Each local data storage unit 26 instance is assigned a uni qds1, qds2. Local data storage unit 26 names typically begin with zero and increment b special character device entry points are provided for each defined local data storage u

As shown in FIG. 3, each local data storage unit 26 appears as a raw disk partition to th data storage unit 26 accepts and handles any request that can be made to a normal ra volume, such as create and mount a file system, or support DBMS table space allocatio

Local data storage units 26 are not shared by the primary computer system 12 and the Rather, data is mirrored across the network 20 from the local data devices 16 to mirror center wants the secondary computer system 14 to assume all activities if the primary application software must reside on both computer systems 12, 14. Applications on the are not executed until the secondary computer system is required to act as the applicat 32 on the secondary computer system 14 are not accessed while the computer network 10 is in normal operation.

Referring to FIGS. 1 and 3, the computer network remote data mirroring system 10 cor installed just above the actual disk device drivers or volume management device driver systems or applications 28. As a result, any disk-based file system supported by Solaris computer network remote data mirroring system 10, as are applications 28 that work d (databases). Advantageously, the device driver 22 for the local data device 16 and the top of the operating system environment, so that the computer network remote data m many commercially available computer systems

The writelog device 18 is accessed only by the device driver 22. All software-based utili writelog device 18 through private I/O control (IOCTL) calls to the device driver 22.

The device driver 22 supports block and special character devices which provide service Typically, block device drivers are limited to transferring blocks of a fixed size and use a between the driver and the user application or file system. The special character device be addressed in units smaller or larger than the device block size. These transfers are p file system or buffer cache and allow the kernel to transfer data directly to or from the unit 26. The device driver 22 requires no modifications to the kernel binaries. FIG. 3 sh driver 22 in a conventional UNIX kernel and its relationship to the local data storage un

As shown in FIG. 3, the device driver 22 performs a data update both to the local data chronologically sequenced journal storage area, or writelog device 18. Once written to t writelog device 18, I/O operation returns control to the application. This delivers to the comparable to simple local disk mirroring.

Considered in more detail, when the device driver 22 receives a call that will modify dat (write or strategy), it places copies of the data on both the local data device 16 and at t 18. Special processing ensures transactional integrity should the primary computer syst writes take place, as will be described shortly. Preferably, the primary computer system controller cards with the local data device 16 configured on one and the writelog device This enables the local data device 16 and the writelog device 18 to be on separate I/O l performance is dramatically improved due to reduced I/O bus contention during the pa embodiment, the local data device 16 and the writelog device 18 are located on separa under separate SCSI controllers for optimal application performance.

The device driver 22 creates the metadata header for the I/O data update that is addec device 18, followed by the data from the update. As described earlier, this header conta the update data, a timestamp, a global sequence number (unique between all writelog sequence number (unique within the current writelog device). These sequence numbers order of the data entries in the writelog device 18 exactly follows the sequence in whicl application.

In accordance with the present invention, a method is provided for ensuring data integ while data updates are occurring in parallel to two data storage devices, such as disk d disks comprise the local data device 16 and, in the embodiment in which the writelog d device, the writelog device disk. The local data device 16 and the writelog device 18 ar through the device driver 22. Preferably, each disk is provided with a disk interface, su

enhance reliability and speed of data updates. Consequently, data is written and thus s simultaneously on both the local data device 16 and the writelog device 18. Therefore, data mirroring system 10 uses the device driver 22 to perform disk updates simultaneo device 16 and to the local, chronologically sequenced journal area, or writelog device 1 written to these two devices, the I/O operation returns control to the application.

The method in accordance with the present invention accommodates any of three condi event of a system crash. The first condition is that the same update data has been stor 16 and in the writelog device 18; the second condition is that the update data was stor failed to be stored in the writelog device; and the third condition is that the update dat device, but failed to be stored on the local data device. In accordance with the method current update data is written to the writelog device 18, while the immediately precedir data device 16. If the primary computer system 12 crashes, upon recovery or re-boot c the two most current data updates in the writelog device 18 are written to the local dat data stored on the local data device is current.

As mentioned earlier, the primary computer system 12 comprises the primary mirror da computer system. The primary mirror daemon 24 monitors the writelog device 18 for u the network 20 in the same order in which it is stored to a receiving remote mirror dae computer system 14, which in turn commits the updates to the mirror device 32. The d journal area of the writelog device 18 on the primary computer system 12 until the rem acknowledgement receipt back to the primary mirror daemon 24 confirming that the da committed to the mirror device 32. This may be thought of as a form of "two-phase cor

The primary mirror daemon 24 is a user-mode background program running on the prir communicates with each secondary computer system 14 in the computer network remc Once a connection is established and authenticated, the primary mirror daemon reques the writelog device 18 through the device driver 22. Data is transferred in chronologica writelog device 18. The primary mirror daemon 24 sends these journaled transactions a secondary computer system 14.

The computer network remote data mirroring system 10 comprises the network 20. Typ networking environment that supports the TCP/IP protocol stack. The computer networ 10 also supports transport technologies including, but not limited to, Ethernet, fast Ethe FDDI.

The secondary computer system 14 comprises at least one mirror device 32 which is sp device (but pertains to both the special character and block mode devices) on the secor which data is mirrored. A mirror device 32 is required on the secondary computer syste 16 on the primary computer system 12. The mirror device 32 must be the same size as corresponding local data device 16. During normal operation of the computer network r 10, the mirror devices 32 contain a coherent, that is, usable, copy of the data stored or 12.

The secondary computer system 14 also comprises the remote mirror daemon 30. The the data updates received from the primary mirror daemon 24 to the mirror device 32 (system. The data updates remain in the writelog device 18 at the primary computer sys daemon 30 sends an acknowledgement receipt back to the primary computer system 1 updates were committed to the mirror device 32.

The remote mirror daemon 30 is a user-mode background program running on the secc remote mirror daemon 30 receives data blocks sent by the primary mirror daemon 24 a associated mirror device 32.

In the exemplary implementation, there are three ways to start the primary mirror dae daemon 30. A first is to launch the daemons 24, 30 on system re-boot automatically by scripts. A second is to execute /opt/QLIXds/bin/launchpmds and /opt/QLIXds/bin/launc command line, or the remote mirror daemon 30 can be automatically started by the ine mirror daemon 24 is launched. A third is to execute /opt/QLIXds/bin/in.pmd and /opt/C from the command line.

The protocol between the primary mirror daemon 24 and the remote mirror daemon 30

communication is established between the primary mirror daemon 24 and the remote n mirror daemon sends an authentication handshake. The information contained within th the connection and instructs the remote mirror daemon 30 to open its copy of the confi information therein, and create internal structures on the secondary computer system : 30 then opens the mirror device 32. To assure the data integrity of these volumes, the exclusively for the remote mirror daemon 30 and cannot be accessed by any other proc daemon is running. The remote mirror daemon 30 receives data updates from the prim these data updates to the mirror device 32, and sends a confirmation that the data upd device back to the primary mirror daemon.

In accordance with the present invention, failure recovery with the primary and remote initiated automatically for certain failures which do not affect the basic operability of th example, the computer network remote data mirroring system 10 is automatically reco of the primary computer system 12 or the secondary computer system 14 or temporar normally established by the network 20. A startup script is provided to automatically st remote data mirroring system 10 when the primary computer system 12 restarts.

Also in accordance with the present invention, the computer network remote data mirrc structured to provide volume grouping, or logical groups 34, as shown in FIG. 5. That is storage units 26 can be configured as a coherent unit, called a logical group 34. In one computer network remote data mirroring system 10 supports up to 512 logical groups : number of local data storage units 26. Placing affiliated local data storage units 26 in th effective way to configure an efficient system. Consequently, data at the primary comp at a plurality of remote computer systems 14, as compared to known architectures whi to a single remote site) data mirroring. Accordingly, the computer network remote data master primary mirror daemon 24 and associated child primary mirror daemons 24A ar remote mirror daemon 30 and associated remote mirror daemons 30A and 30B, to proc graphical user interface is preferably provided by the computer network remote data m configuring the logical groups 34, as well as for monitoring performance of the remote

Considered in more detail, FIG. 5 illustrates the relationship between local data storage primary mirror daemons 24, 24A, 24B, and remote mirror daemons 30, 30A, 30B. It is local data devices 16 configured within a logical group 34 to share a writelog device 18 but this practice is heavily discouraged due to the excessive performance penalty it imp operates with its own independent primary mirror daemon/remote mirror daemon pair, daemon 24A and remote mirror daemon 30A.

There are various reasons to have several logical groups 34 defined. For example, som databases, may work with a number of disk devices at the same time. It is preferable t maintained, not only within a local data storage unit 26, but also between local data stc no more up to date than any other. In such a situation, chronological coherency of I/O organizing the local data storage units 26 into a logical group 34. The logical groups 34 data storage units 26 to ensure chronological coherency is enforced between the memt computer system 12 go out of service at any point during network transfers, the mirror computer system 14 for a logical group 34 will be current to a specific point in time, all cleanly to the secondary computer system. Also, individual logical groups 34 can be tar secondary computer systems 14, thus creating a one-to-many configuration. Additional independent network connections to the secondary computer system 14, thus creating greater than that of any single network connection. Furthermore, the failure of one logi operations of any other logical groups. The aggregate data rate is used in determining sustain the flow of data from the logical group 34.

The logical groups 34 represent the highest level of organization for the computer netw system 10. Configuration files (e.g., /etc/opt/QLIXds/dsgrp000.cfg) are defined for eac primary and secondary computer systems 12, 14, writelog device extension pools 18A, parameters, throttles, and configurations of the member local data storage units 26. FI between the logical groups 34 on the primary computer system 12 and the secondary c member local data storage units 26, and the primary and remote mirroring daemons 2 has its own set of daemon processes, the failure of a single logical group will not impac other logical groups. An additional feature of the computer network remote data mirror logical group 34 may be independently targeted to a secondary computer system 14. T channels between the primary and secondary computer systems 12, 14 may be exploit

groups 34, or that the primary computer system 12 may target multiple secondary com data mirroring.

As described earlier, a writelog device extension is specified as a special character disk overflow of the current writelog device 18 is imminent. The primary mirror daemon 24 . device extensions from the writelog device extension pool 18A when any local data stor group 34 reaches the threshold for overflow or under utilization as set by throttles in th writelog device extensions are not utilized by default; that is, writelog device extension logical group definitions. They are a safeguard available against writelog device overflo the configuration file and these devices must be included in the writelog device extensi writelog device extensions are stored in the writelog device extension pool 18A for eacl the pool are available to any local data storage unit 26 in the logical group 34. In one e individual local data storage unit 26 can have a total of sixteen writelogs (one main anc at any point in time.

In the embodiment in which the computer network remote data mirroring system 10 cc primary mirror daemon 24 looks in the /etc/opt/QLIXds directory and creates a child pr for each configuration file that it finds. A configuration file exists for each logical group group 34 has its own primary mirror daemon process. In the embodiment which compr primary mirror daemon includes the child processes. The term master primary mirror d main dispatching daemon, that is, the primary mirror daemon 24. The master primary i the child primary mirror daemon processes and relaunches them should they fail unexp a temporary network failure.

Each primary mirror daemon 24A, 24B reads its assigned configuration file. The primar opens a remote connection to the port number of the remote mirror daemon 30A, 30B system 14 given in the configuration file (defaultport: 575). Since each logical group 3 process, each logical group may have a connection to a distinct and separate secondar

As with the master primary mirror daemon 24, in the embodiment in which the comput mirroring system 10 comprises logical groups 34, there is a master remote mirror daen network 20 for new connections and creates a child remote mirror daemon process for comprises logical groups 34, the term remote mirror daemon includes the child process mirror daemon is used to identify the main dispatching daemon, that is, the remote mi

The set of daemon processes, that is, the primary mirror daemon 24A, 24B on the prim the remote mirror daemon 30A, 30B on the secondary computer system 14, are used t updates from the writelog devices 18 of a logical group 34 across the network 20 onto t of the secondary computer system 14. These daemons 24A, 24B, 30A, 30B create a TC known socket port (defaultport: 575) to effect this transfer of data updates. The protoc minimal.

Advantageously, whether configured with or without logical groups 34, the computer ne system 10 operates over different network configurations and is compatible with many storage devices which comprise the local data device 16, writelog device 18, and mirro network remote data mirroring system 10 operates with all disk subsystems supported 2.X environment, each disk is partitioned and accessed in the manner described in the AnswerBook The computer network remote data mirroring system 10 allows the user tc database or file system data by simply incorporating the devices or volumes on which t computer network remote data mirroring system 10. Disks do not need to be repartitio do not need to be re-initialized, and data does not need to be exported/imported.

In the exemplary implementation, after the installation and configuration of the comput mirroring system 10, file system mount tables or applications taking advantage of raw reference the local data storage unit 26 created (e.g., /dev/dsk/qds13) rather than the (e.g., /dev/dsk/e3t2d0s4) or managed volume (e.g., /dev/vx/dsk/vol3)to have all I/O t the secondary computer system 14. This installation can take place without modificatio backup/restore or conversion activity is required of the data set.

In accordance with the present invention, various throttles are provided. Throttles are a automate the administration of the computer network remote data mirroring system 1C mechanism that keeps a computer system operating within a range defined by the thro

the configuration and alerting the user to system trends or problems. The primary mirr throttles periodically and performs the actions specified by them.

Considered in more detail, throttles are user-defined tests and actions evaluated by the periodically. Throttles have a multifold purpose: to limit system and network resource c network remote data mirroring system 10; to deal with pending writelog device overflo dynamically; to notify system and network administrators of problems; and to execute

Throttles are defined in the logical group configuration file. An unlimited number of thrc logical group 34, and each throttle may have up to sixteen actions that are executed if Throttles may be defined to evaluate only between certain times of day, making it poss network remote data mirroring system 10 around business needs, such as not utilizing during normal business hours.

Throttles are not a general purpose macro language, but a very efficient set of program impact the performance of the primary mirror daemon network transfers when execute computer network remote data mirroring system 10 is configured with one or more logi sophisticated testing against the following run-time determined measurements: the KB 20 per second for the logical group 34; the percent of the writelog device 18 in use for and the percent of central processing unit (CPU) resources that the child primary mirro consuming. Actions that a throttle may invoke include: set, increment, or decrement a performed prior to each data entry transmission; set, increment, or decrement a sleep local data storage unit 26 will wait prior to returning control to applications after an upc device to a writelog device 18 to increase the size of the writelog device (up to sixteen exemplary implementation); free a writelog device extension disk device to go back to pool 18A until needed again; log a message to syslog; write a message to the system c to the user; and execute an arbitrary shell command (built-in, script, or external progra failure is detected, the on-call operator could be paged. Once the failure is cleared, a m could update the status display.

Throttles are preferably configured using a graphical user interface provided by the qds FIG. 6, a "Throttles" tab brings up a screen that enables the user to define throttles for The throttles screen is divided into two sections. One section is a "Throttles Defined" se portion of the screen. It provides the elements required to create a throttle. It also lists for the given logical group 34.

The second section is an "Actions for Throttle" section which appears in the lower portic the ACTIONLIST, identifies selectable actions to be carried out when the throttle evalua implementation, each throttle may have up to sixteen actions in the ACTIONLIST. The u by clicking the down arrow to the right of the "Actions for Throttle" field.

The buttons at the bottom of the screen, namely, "New," "Commit," "Delete," and "Res while those located at the center of the screen apply only to the ACTION currently bein clicks on the arrow to the right of the "Throttles Defined" field to display a list of all exis given logical group 34.

In order to create a throttle, the following steps are performed. First, the user clicks on appears at the bottom of the screen. Second, the user enters a "From" and a "To" time applicable. This identifies the time span during which the throttle is active. If the throttl user simply enters "--" in both the "From" and "To" fields. Third, the user selects a "Thr by initially clicking on the down arrow next to the "Throttle Test" field and then choosin ("netkbps," "pctcpu," or "pctwl") to determine which system element is regulated by th The three options determine which system component, namely, network bandwidth, CP is being controlled. These options are as follows: "netkbps" is the KB per second being from the primary mirror daemon 24 to the remote mirror daemon 30; "pctcpu" is the p consumed by the primary mirror daemon; and "pctwl" is the percentage of the writelog writelog device in the logical group 34. Fourth, the user chooses a relational or transitic pulldown menu in the center of the screen, as shown in FIG. 8. Fifth, the user enters ar field for comparison to the actual system component usage. This completes the creatio logical group 34.

In order to create the ACTIONLIST for the new throttle, the user performs the following

clicks on the "New" button in the "Actions for Throttle" section of the screen. Second, tl the "Do What" pulldown menu, as shown in FIG. 9. For example, "sleep" causes the pri given logical group 34 to sleep a preselected number of microseconds after each transn mirror daemon 30 when the criteria outlined in the throttle definition are met. Accordin limit consumption of network and CPU resources. Third, the user enters any applicable field to preselect the extent of the selected action. Fourth, the user clicks on the "Comn center of the screen to elect the ACTION. If the user wants to assign more ACTIONs to repeats the first four steps used to create the ACTIONLIST. In one exemplary implemer every ten seconds by default.

In accordance with the present invention, the computer network remote data mirroring bandwidth throttling. Bandwidth throttling enables a predetermined portion of the band assigned to remote data mirroring depending on the time of day or other criteria. Accor network consumption during prime hours of operation and automatically increase band

Considered in more detail the computer network remote data mirroring system 10 enat performance. That is, the computer network remote data mirroring system 10 allows tl network bandwidth available for the data replication process. By doing so, the user can services during peak times and maximize replication when more bandwidth is available.

To optimize performance, several methods can be used to minimize processing and I/O network throughput. The computer network remote data mirroring system 10 includes user to fine-tune the rate at which the primary mirror daemon 24 transfers data, based dynamic state of the application. Throttles allow the user to limit the network bandwidtl the computer network remote data mirroring system 10. Furthermore, the user can aut based on the time of day.

For example, if the user network connection is congested, the user may choose to slow computer network remote data mirroring system 10 during peak hours (e.g., 8:00 AM t the restriction after hours. The dynamic controls of the computer network remote data user to define how much and when computer system/network resources are used for re

An example of network bandwidth throttling is illustrated in FIG. 10. In the example, th network bandwidth consumption. The first two throttles deal with maintaining usage of point. Note that "sleep" is incremented by 15,000 microseconds if usage exceeds 200 K continues to increase and exceeds 300 KB per second, "sleep" is incremented by 5,000 throttle evaluates true. The remaining throttles focus on maintaining network usage. If is decremented continuously until it reaches zero.

In operation, the preferred configuration of the computer network remote data mirrorin accumulation of data updates in the writelog device 18 to occur independently of the tr to the secondary computer system 14. This dissociation is termed asynchronous accum asynchronous mode. The advantage of such a configuration is that applications realize while the daemon processes can exploit network bandwidth optimally. The asynchronou mode for the embodiment in which the computer network remote data mirroring syster groups 34. In the asynchronous mode, the journaling of data updates to the writelog d the primary mirror daemon process will transmit the data updates to the secondary cor

Alternatively, the computer network remote data mirroring system 10 can be configure synchronous mode does not return control to an application until after a data update ha local data device 16 on the primary computer system 12 and the mirror device 32 on tl 14. In the synchronous mode, the mirror device 32 is an exact copy of the local data de UFS file systems on top of a device defined in synchronous mode will continue to perfor The synchronous mode is an operating mode that requires that the primary mirror daer the secondary computer system 14 and receive confirmation that it was committed to t allowing the device driver 22 to return control to the application being executed on the The synchronous mode provides the user of extremely critical, reliable data a way to er disk synchronously on the primary computer system 12 is stored on the secondary com primary is allowed to continue with its operations. Many banking and billing systems fal UNIX is not reliable in this sense. It delays writes to the disk to improve performance. T asynchronous manner. Given some of the constraints and limitations of the UFS implem necessary to process UFS asynchronous writes asynchronously, as if in the asynchronou

synchronous mode increases latency to disk of metadata. This has the effect of decreas systems mounted on a local data storage unit 26 in the synchronous mode. For these r use the synchronous mode on a UFS file system. On the other hand, in one exemplary i systems honor the synchronous mode if mounted with a command -o mincache=dsync.

In another alternative, the computer network remote data mirroring system 10 can be synchronous mode. The near synchronous mode is a middle ground between asynchror behaviors. In the near synchronous mode, data updates are allowed to accumulate in tl asynchronously until a tunable number of data entries has accumulated, at which time are blocked until the number of entries in the writelog device falls below this tunable lir reduces the performance penalty found in the synchronous mode, yet adds confidence device 32 on the secondary computer system 14 is no more than n disk updates behinc 12. The near synchronous mode is a relaxation of the synchronous mode to allow up to in the writelog device 18 asynchronously before blocking application I/O until the writel entries.

In the exemplary implementation, the computer network remote data mirroring system operating modes. These are described below.

Most of the following modes are not exclusive. That is, the computer network remote d operate in one or more of these modes concurrently.

A setup mode is the default mode for the computer network remote data mirroring syst installed. The setup mode indicates that the local data storage unit 26 has not been cre remote data mirroring system 10 is not in operation. It is in this mode that configuratio running add_drv qds (which establishes the local data storage unit 26). The primary mi mirror daemon 30 are not running in this mode.

In an accumulating mode, the computer network remote data mirroring system 10 is in directed to the local data storage unit 26. Modifications to local data are being journale the primary mirror daemon 24 is not active and is not removing these. The accumulatir failed network connection or explicit halting of the primary mirror daemon 24 (killpmds device 18 continues to fill while entries are not being removed. This eventually causes t overflow. If a logical group 34 is configured for the synchronous mode or the near sync logical group in accumulating mode will block I/O updates from the applications.

A connected mode indicates that the software which comprises the computer network r is installed, read/write requests are being handled by the local data storage unit 26, an transferred to the mirror device 32 on the secondary computer system 14 through the the remote mirror daemon 30. Modifications to local data are being journaled to the wri primary mirror daemon 24 is actively transferring entries from the writelog device 18 tc on the secondary computer system 14. The remote mirror daemon 30 commits the dat connection to the mirror device 32 on the secondary computer system 14. This is the o data mirroring is accomplished and in which ongoing mirroring and a coherent copy of computer system 14.

The computer network remote data mirroring system 10 is placed into a bypass mode t required arguments or automatically when a writelog device overflow occurs. When the mirroring system 10 is in the bypass mode, the software which comprises the computer system is installed, and reads/writes are being done to the local data storage unit 26, t reading from and writing to the local data device 16 only. No journaling of data modific writelog device 18. Updates are not being transferred to the secondary computer syste operating state into which a local data storage unit 26 moves automatically when a writ overflows. Moving into the bypass mode results in the secondary computer system 14 t the primary computer system 12 and requires a refresh operation to re-establish synch systems.

A refresh mode is a background operation used to create an initial mirror or to synchro system 14 with the primary computer system 12. The computer network remote data n the refresh mode by typing launchrefresh with the required arguments. A refresh cause storage unit 26 to be moved across to the secondary computer system 14 by reading fr writing the sectors to the writelog device 18. This can be performed while other I/O act

local data storage unit 26 as well. The refresh mode is halted automatically when the re normal operation commences. While the computer network remote data mirroring syste the data on the secondary computer system 14 is in an incoherent state. The data on tl 14 preferably cannot be used if a failure occurs during the refresh process. Coherency i is complete. For this reason, it may be necessary to back up the secondary computer s computer network remote data mirroring system 10 in the refresh mode.

A backfresh mode is an operation used to synchronize the primary computer system 12 system 14. A backfresh operation causes all the sectors of the mirror device 32 to be m the corresponding local data device 16. The primary mirror daemon 24 is placed into th launchbackfresh command with the required arguments. Once in the backfresh mode, t requests mirror device sectors from the remote mirror daemon 30 and writes the returr device 16 for all target data devices. When the backfresh operation completes, the prim automatically restored to the state it was in, typically the connected mode or the accun launchbackfresh command was issued. While the backfresh operation is running, the lo off-line to applications, because the backfresh operation requires exclusive access to th the backfresh mode, the data on the local data device 16 is in an incoherent state. The if a failure occurs during the backfresh operation. Coherency is achieved only after the

During normal operation, the primary mirror daemon 24 fetches a large block of entries one time and sends the data entries, one at a time, across the network 20 to the remot advantage of having the primary mirror daemon 24 obtain a number of data entries at head contention in the embodiment in which the writelog device 18 comprises a disk dr the disk head back and forth between the head and the tail of the writelog device, whic performance. If there are only one or two data entries in the writelog device 18, this re daemon 24 is satisfied entirely out of the in-memory cache of the device driver 22, and data entries in the writelog device 18 remain in the writelog device until the primary mi acknowledgement for the data entries from the remote mirror daemon 30, meaning the mirror device 32. When this occurs, the primary mirror daemon 24 informs the device d pointer of the writelog device 18, and that space used by the transmitted mirrored entr

The writelog device 18 maintains metadata describing where the head and the tail are l embodiment in which the writelog device comprises a disk drive device. This is actually of-date guess. The primary mirror daemon 24 periodically requests that the device driv the current values based on either a tunable timeout being reached or after m data ent remote mirror daemon 30, whichever is encountered first. During the startup/recovery remote data mirroring system 10, a qdswlscan utility uses these head and tail locations writelog device 18 and locate the actual head and tail locations. Maintaining a recent lo metadata area reduces the time required for this scan to complete. Timestamps and se evaluating which data entries actually constitute the head and tail of the writelog devic

It is important to note that the data from an update is written to both the local data de 18. One perception might be that this is inefficient from a storage standpoint, that is, tl on the system, rather than simply using offset and length in the writelog device entry h read an entry and transmit it across the network 20. This brings out a design feature of data mirroring system 10, namely, chronological coherency of data. Chronological cohe device 32 on the secondary computer system 14 not only receives all data from update system 12, but that the mirror device always is in a usable state and contains a faithful particular instant in time.

At any given moment, the writelog device 18 may contain tens or hundreds of updates were to simply use the offset and length stored in the entry header in the writelog devi data device 16, one may instead be fetching data that was modified after the transactic one is sending "out of order" data, meaning the mirror device 32 has information that i 11 shows the consequence of using offsets and lengths to read the entry from the data storing a copy of the data in the writelog device 18.

In the case in which the computer network remote data mirroring system 10 is configu groups 34, when the master primary mirror daemon 24 begins execution, it looks for lo in /etc/opt/QLIXds. For each logical group configuration file it finds that is defined for tl role, it creates a child process for that logical group 34. After all child processes have b

mirror daemon process waits for any child process terminations. The master primary m the cause of the child process termination and will relaunch the child if the cause was d as a loss of the network link

The child primary mirror daemon process concerns itself only with the logical group con by the master primary mirror daemon 24. It reads this file, creates the necessary data configured devices, then attempts to create a connection to the secondary computer sy made to a master remote mirror daemon process on the secondary computer system 1 mirror daemon 30 receives a connection, it creates a child remote mirror daemon proce private channel of network communications with the primary mirror daemon process. T 24A, 24B sends an authentication handshake to the child remote mirror daemon 30A, 3 child remote mirror daemon process which configuration file is used for the logical grou uniquely identifies the primary computer system 12. The child remote mirror daemon 3 configuration file and various system information to verify the identity of the primary cc continues if this is an appropriate connection or terminates if the primary computer sys the connection. The child primary mirror daemon 24A, 24B then sends a mapping for e has in its configuration file to the child remote mirror daemon process. The child remot verifies the local data storage unit 26 mapping against its configuration file and returns (ACK) message and the next expected sequence number for the device or an error (ERI mismatch and then terminates. Once all mappings are in place, the child primary mirro fills internal buffers with data entries from the associated writelog devices 18. These da their global sequence number ordering, ensuring that chronological coherency is mainta data storage units 26 of the logical group 34. When the last data entry of the internal b unit 26 is being prepared to be sent, a flag is set in the header requesting an acknowle mirror daemon 30A, 30B once the data of the entry has been committed to the mirror c the protocol overhead thin in that each data entry transferred does not require an expli remote mirror daemon 30A, 30B. Once this ACK message has been received from the c process, the child primary mirror daemon 24A, 24B has the device driver 22 advance th over all of the committed data entries.

The child primary mirror daemons 24A, 24B periodically update a performance tracking with the same name as the configuration file (e.g., dsgrp000.prf). This file is ASCII and row corresponds to the time of the last update of the performance file. New data entrie files.

A number of performance metrics are issued for each local data storage unit 26 since th performance metrics preferably include the total KB per second (header plus data) sent system 14; the data KB per second sent to the secondary computer system; the data e secondary computer system; the number of data entries in the writelog device 18 awai sectors in use in the writelog device in the embodiment in which the writelog device co percent of the writelog device in use; and the age of the oldest data entry in the writelc performance files are allowed to grow to a configurable limit. In one exemplary implem has a default of 64 KB. One previous generation of these performance files is retained. implementation, a graphical utility program qdsperftool, is provided to instantly view ar information in real-time, presentation quality, configurable charts.

Finally, the exemplary implementation of the computer network remote data mirroring commands. The following describes various utility commands to manage the computer system 10 environment.

qdswlinit is a utility command to initialize one or more writelog devices 18 to show it er is preferably done prior to the first use of a writelog device 18.

qdswlscan is a utility command to associate and register with the device driver 22, for c units 26, the writelog device 18 and its corresponding local data device 16. This comma driver 22 of writelog device extensions, sequence numbers, and the head and tail of the required activity prior to using the local data storage unit 26.

qdsbypass is a utility command which turns off or on the bypass mode for one or more When in the bypass mode, updates are not added to the writelog device 18. When in th broken, as data updates are not be transferred to the secondary computer system 14. I modify the local data device 16. A writelog device overflow during run time preferably c

daemon 24 to automatically place all local data storage units 26 in the logical group 34

qdsrefresh is a utility command for one or more local data storage units 26, which copi from the local data device 16 to the writelog device 18, and then over the network 20 t secondary computer system 14. This command is used to create an initial mirror of dat system 14 or to refresh the secondary after a writelog device overflow. This utility will r 18, as it constantly monitors the availability of writelog device resources. Normal I/O ca storage unit 26 when qdsrefresh is running without threat of data loss or corruption. It qdsrefresh cycle, the mirror device 32 is placed into an incoherent state and is not usat primary computer system 12 go out of service during the process.

qdsinfo is a utility command which prints out the state and metrics for one or more writ device driver 22. This utility is executed only on the primary computer system 12. The ASCII report for one or more local data storage units 26 from the perspective of the de device 18. The qds info command indicates if the writelog device 18 has been placed in the bypass mode or the refresh mode. It also shows performance metrics specific to th

qdsconfigtool is a graphical user interface utility for viewing, editing, or defining logical including primary and secondary computer systems 12, 14, tunable primary mirror dae extension pools 18A, local data storage units 26, and throttles.

qdsperftool is a graphical real-time charting tool for displaying performance data for th mirroring system 10. The user employs qdsperftool to display charts of various perform network remote data mirroring system 10. For example, FIG. 12 illustrates a chart gen Additionally, the user can display multiple charts at one time, modify the charts, delete enables the user to observe performance of the computer network remote data mirrori shows trends.

qdsmonitortool is a utility that provides a comprehensive picture of activity and state in network remote data mirroring system 10. This utility is executed only on the primary qdsmonitortool is a graphical user interface utility that shows in a single window the fol status message area; error messages and warnings from the monitored primary compu computer systems 14 associated with the primary computer system; logical groups 34 primary mirror daemon 24 and the state of the secondary mirror daemon 30; local data and their activity modes; writelog device states and usages; and a notification control p

qdsdevinfo is a convenience utility that returns the major/minor numbers for any disk c in sectors (used by qdsconfigtool).

qdshostinfo is a convenience utility that returns the networked hostnames and IP addre qdshostinfo also returns the host identification if executed on the computer system in q

qdsrmdreco is a recovery utility for the secondary computer system 14, that commits a mirror device 32.

launchpmds, launchrmds, and launchrefresh are shell scripts that are the preferred way secondary mirror daemons 24, 30. These shell scripts perform sanity checks before run run the daemons under a "nohup" so that they will not terminate when the parent wind

killpmds, killrmds, and killrefresh are shell scripts that conveniently terminate all daem

Finally, in.pmd and in.rmd are utility commands to render the primary mirror daemon 2 mirror daemon 30 executable, respectively.

In accordance with the foregoing description, the computer network remote data mirro disaster recovery operation by maintaining up-to-date copies of the disk storage at a re using a second copy and a different server, the business-critical resource at the primar available during a required backup process. Accordingly, the system downtime needed reduced. Additionally, when moving a data center to a new location or upgrading equip computer network remote data mirroring system 10 may be used to keep the primary s being migrated.

The computer network remote data mirroring system 10 is implemented both locally an particularly useful when it is incorporated into a server fail-over computer system. A se the problem of a system CPU which goes out of service, as compared to a disk going ou over system is described in commonly-owned co-pending U.S. patent application Series entitled "SERVER FAIL-OVER SYSTEM," filed on Nov. 10, 1997, which is incorporated in reference.

In summary, the computer network remote data mirroring system 10 provides high-pe protection for networked computer systems, such as UNIX-based computer systems. Th data mirroring system 10 is compatible with many storage devices, file systems, and vo network remote data mirroring system 10 provides continuous network data mirroring a replication while ensuring fast application response time without risking application ava remote data mirroring system 10 is preferably configured to automatically recover from computer system 12, secondary computer system 14, and network 20. The computer n system 10 comprises an intuitive graphical user interface to ensure ease of configuratio to provide management of network resources, while throttles enable triggered events to overall process. Nevertheless, the computer network remote data mirroring system 10

The computer network remote data mirroring system 10 protects critical business infor data from disks on a local system to a user-specified target system that may be locatec simply across the hall. Unlike solutions that only replicate a database, the computer net system 10 enables the user to mirror any disk partition or volume. And, unlike hardwar the same storage hardware, typically at the same site, the computer network remote d the user to replicate data to a remote site, without having to replicate the hardware en hardware failure or a natural disaster at the primary location, the computer network re provides an open-systems, real-time data mirroring solution to help ensure that vital da

The computer network remote data mirroring system 10 enables the user to efficiently mirroring process. Using its powerful dynamic controls and graphical and text monitorir when and to what degree system or network resources are used. For example, the use consumption during prime hours of operation and automatically increase bandwidth use

The computer network remote data mirroring system 10 comprises intuitive tools to mo replication process, as well as respond to specific occurrences. The software includes bo user interfaces. The graphical interface uses standard X libraries that are readily familia

Snapshot and continuous monitoring tools give the user instant insight into the status o the system and network resources are being used. Performance monitoring provides th operation in a graphical format.

In the foregoing specification, the present invention has been described with reference It will, however, be evident that various modifications and changes may be made there broader spirit and scope of the invention. One contemplated modification is to provide s checkpointing of mirrored data during active data updates, as well as to provide an add during incoherent data transfer periods (e.g., during synchronizations after overflows ir writelog device 18 of the computer network remote data mirroring system 10 comprise disk). Another contemplated modification is optimization for high data rate throughput includes: continuous data streaming, simple compression of data streams, banding of n logical connection with additive bandwidth increases, and smart synchronization of prim Continuous network data streaming would yield data transfer rates approaching the ma seen with high performance networks (comparable to ftp speeds). User activation of da effective network data transfer rates exceed any other LAN-based network transfer me network banding would allow network bandwidth to be added to a network connection b network paths between primary and secondary systems. Smart data synchronization (r transfer only changed data between primary and secondary sites, tremendously reducir contemplated change is that throttle evaluation and performance data collection can be secondary mirror daemons 24, 30 and placed in a separate daemon executed on the pr systems 12, 14.

Furthermore, while one exemplary implementation has been described with respect to a invention is also applicable to other operating environments, such as Microsoft Windows

the principles of the present invention apply to Octopus Datastar real-time data protect networks and OctopusDP real-time data protection software for Windows 95 and Windo are commercially available from Qualix Group, Inc., located in San Mateo, Calif., and wl herein in their entirety by this reference. The specification and drawings are, accordingl illustrative rather than in a restrictive sense. The present invention should not be const embodiments and examples, but rather construed according to the following claims.

<- Previous Patent (Second operation in internal peripheral ..) | Next Patent (Input/output controller providing preven..) ->

Patent RSS Feeds

Tom Sheldon's

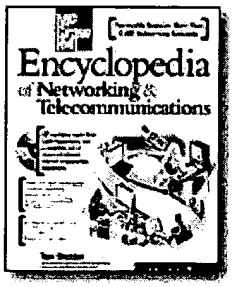
Linktionary™.com

Networking Defined and Hyperlinked

Site home page (news and notices)

Get alerts when Linktionary is updated

Book updates and addendums



Get info about the Encyclopedia of Networking and Telecommunicatons, 3rd edition (2001)

Mirroring

Related Entries Web Links New/Updated Information

Find! Topic Indexes

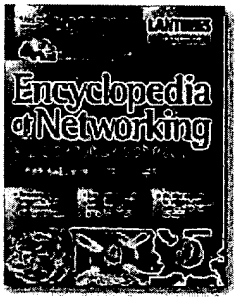
Search Linktionary (powered by FreeFind)

Note: Many topics at this site are reduced versions of the text in "The Encyclopedia of Networking and Telecommunications." Search results will not be as extensive as a search of the book's CD-ROM.

Mirroring is the process that exactly duplicates information in one location to another. It can be done locally-say, between disk drives in the same system-or globally, such as when information on a server is duplicated to a server in other parts of the world. Many Web sites mirror their content to other servers, bringing information closer to users, and reducing the distance and number of router hops that data must travel to get to a user. Content distribution is all about getting Web-based information closer to users.

Most network operating systems and many desktop operating systems support disk mirroring. In this type of mirroring, stored data on a primary drive is continuously copied to a second storage device in real time so that both devices hold the same information. Mirroring is a form of fault tolerance that protects data from equipment failure. There are several different types of mirroring, as described here and pictured in Figure M-2.

- **Mirroring** Data is copied from on-disk controller (channel) to two disk drivers. If one drive fails, the other is still operational.
- **Duplexing** Data is duplicated over two disk channels and



stored on two drives. This method extends fault tolerance to the controller.

- **Server duplexing** This method provides fault tolerance by duplicating the entire file server. If one server fails, the other provides continuous service to users. For example, Novell's *System Fault Tolerance* provides server duplexing.
- **Replication** A strategy of duplicating critical files and directories from a server at one location to a server at another location to make that information more accessible to users at the remote location and also to provide redundancy and backup. See "Redundancy" and "Replication."
- **Clustering** A cluster is a group of servers that share access to the same resources and service clients equally. Should one of the servers go down, the others take up the processing load. Clustered servers may access the same disk systems, which may be mirrored or in a RAID configuration. See "Clustering."
- **Mirror site** A mirror site is a duplicate data center, located at another site, that contains duplicate systems and data. The duplicate data center should go into operation as the primary site if the master data center site fails for any reason. Companies running mission-critical applications will often create mirrored sites. See "Data Center Design."

Many organizations today outsource in order to obtain the benefits and features of the mirror techniques described earlier. They may also choose to co-locate duplicate equipment at public Internet data centers. Outsourcing and/or co-location allows organizations to gain all the management and security benefits of Internet data centers. See "Outsourcing."

Content distribution is an advanced, highly managed form of mirroring. It improves performance of Web sites by placing copies of content at caching servers close to users. These servers are placed at ISP sites and Internet data centers by companies such as Akamai. One can imagine a ring of content distribution devices surrounding the Internet. When users access Web sites, the content they are looking for is likely to be cached in a content distribution server near them. See "Content Distribution." Also see "Web Caching."

A quantitative analysis of cache policies for scalable network file systems

Full text Pdf (1.42 MB)

Source **Joint International Conference on Measurement and Modeling of Computer Systems** archive
Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems table of contents
Nashville, Tennessee, United States
Pages: 150 - 160
Year of Publication: 1994
ISBN:0-89791-659-X
Also published in ...

Authors
Michael D. Dahlin Univ. of California at Berkeley, Berkeley
Clifford J. Mather Univ. of California at Berkeley, Berkeley
Randolph Y. Wang Univ. of California at Berkeley, Berkeley
Thomas E. Anderson Univ. of California at Berkeley, Berkeley
David A. Patterson Univ. of California at Berkeley, Berkeley

Sponsor SIGMETRICS: ACM Special Interest Group on Measurement and Evaluation

Publisher ACM Press New York, NY, USA

DOI Bookmark: Use this link to bookmark this Article: http://doi.acm.org/10.1145/183018.183034
What is a DOI?

↑ ABSTRACT

Current network file system protocols rely heavily on a central server to coordinate file activity among client workstations. This central server can become a bottleneck that limits scalability for environments with large numbers of clients. In central server systems such as NFS and AFS, all client writes, cache misses, and coherence messages are handled by the server. To keep up with this workload, expensive server machines are needed, configured with high-performance CPUs, memory systems, and I/O channels. Since the server stores all data, it must be physically capable of connecting to many disks. This reliance on a central server also makes current systems inappropriate for wide area network use where the network bandwidth to the server may be limited.In this paper, we investigate the quantitative performance effect of moving as many of the server responsibilities as possible to client workstations to reduce the need for high-performance server machines. We have devised a cache protocol in which all data reside on clients and all data transfers proceed directly from client to client. The server is used only to coordinate these data transfers. This protocol is being incorporated as part of our experimental file system, xFS. We present results from a trace-driven simulation study of the protocol using traces from a 237 client NFS installation. We find that the xFS protocol reduces server load by more than a factor of six compared to AFS without significantly affecting response time or file availability.

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

Arch86 James Archibald , Jean-Loup Baer, Cache coherence protocols: evaluation using a multiprocessor simulation model, ACM Transactions on Computer Systems (TOCS), v.4 n.4, p.273-298, Nov. 1986

Bake91 Mary G. Baker , John H. Hartman , Michael D. Kupfer , Ken W. Shirriff , John K. Ousterhout, Measurements of a distributed file system, Proceedings of the thirteenth ACM symposium on Operating systems principles, p.198-212, October 13-16, 1991, Pacific Grove, California, United States

Birr93 Andrew D. Birrell, Andy Hlsgen, Chuck Jerlan, Timothy Mann, and Garett Swart. The Echo Distributed File System. Technical Report 111, Digital Equipment Corp. Systems Research Center, 1993

Blaz91 Matt Blaze and Rafael Alonso. Long-Term Caching Strategies for Very Large Distributed File Systems. In Proc. of. the Summer 1991 USENIX, pages 3-15, June 1991.

Blaz92 Matt Blaze and Rafael Alonzo Dynam,c H~erarchlcal Caching in Large-Scale Distributed File Systems. In Proc. oj the 12th Internattonal Conf. on Dtatributed Computing System.s, pages 521-528, June 1992.

Blaz93 Matthew Addison Blaze, Caching in large-scale distributed file systems, Princeton University, Princeton, NJ, 1993

Chen93 Peter M. Chen , David A. Patterson, A new approach to I/O performance evaluation: self-scaling I/O benchmarks, predicted I/O performance, Proceedings of the 1993 ACM SIGMETRICS conference on Measurement and modeling of computer systems, p.1-12, May 10-14, 1993, Santa Clara, California, United States

Coyn93 Robert A. Coyne and Harry Hulen. An Introduction to the Mass Storage System Reference Model, Version 5. In Twelfth IEEE Symposium on Mass Storage $3 ~tems, pages 47-53, April 1993.

Gold93 Jonathan S, Goldlck, Kathy Bennlnger, Woody Brown, Christopher Kirby, Christopher Maher, Daniel S, Nydick, and Bill Zumach. An AFS-Based Supercomputing Environment. In Twelfth IEEE Symposzum on Mass Storage Systems, pages 127- 132, April 1993.

Hage92 Erik Hagersten , Anders Landin , Seif Haridi, DDM: A Cache-Only Memory Architecture, Computer, v.25 n.9, p.44-54, September 1992

Henn90 David A. Patterson , John L. Hennessy, Computer architecture: a quantitative approach, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1990

Hitz90 David Hitz, Guy Harris, James K. Lau, and Allan M. Schwartz. Using UNIX as One Component of a Lightweight Distributed Kernel for Multiprocessor File Servers. In Proc. of the Winter 1990 USENIX, pages 285-296, 1990.

Howa88 John H. Howard , Michael L. Kazar , Sherri G. Menees , David A. Nichols , M. Satyanarayanan , Robert N. Sidebotham , Michael J. West, Scale and performance in a distributed file system, ACM Transactions on Computer Systems (TOCS), v.6 n.1, p.51-81, Feb. 1988

Katz91 Randy H. Katz , Thomas E. Anderson , John K. Ousterhout , David A. Patterson, Rob-line

Storage: Low Latency, High Capacity, University of California at Berkeley, Berkeley, CA, 1991

Kist92 James J. Kistler , M. Satyanarayanan, Disconnected operation in the Coda File System, ACM Transactions on Computer Systems (TOCS), v.10 n.1, p.3-25, Feb. 1992

Lazo86 Edward D. Lazowska , John Zahorjan , David R. Cheriton , Willy Zwaenepoel, File access performance of diskless workstations, ACM Transactions on Computer Systems (TOCS), v.4 n.3, p.238-268, Aug. 1986

Leno90 Daniel Lenoski , James Laudon , Kourosh Gharachorloo , Anoop Gupta , John Hennessy, The directory-based cache coherence protocol for the DASH multiprocessor, Proceedings of the 17th annual international symposium on Computer Architecture, p.148-159, May 28-31, 1990, Seattle, Washington, United States

Lisk91 Barbara Liskov , Sanjay Ghemawat , Robert Gruber , Paul Johnson , Liuba Shrira, Replication in the harp file system, Proceedings of the thirteenth ACM symposium on Operating systems principles, p.226-238, October 13-16, 1991, Pacific Grove, California, United States

Mogu87 J. Mogul , R. Rashid , M. Accetta, The packer filter: an efficient mechanism for user-level network code, Proceedings of the eleventh ACM Symposium on Operating systems principles, p.39-51, November 08-11, 1987, Austin, Texas, United States

Munt92 D. Muntz and P. Honeyman. Multi-level Caching in D~stnbuted File Systems or Your cache ain't nuthin' but trash. In Proc. oj the Winter 1992 USENIX, pages 305-313, January 1992.

Nels88 Michael N. Nelson , Brent B. Welch , John K. Ousterhout, Caching in the Sprite network file system, ACM Transactions on Computer Systems (TOCS), v.6 n.1, p.134-154, Feb. 1988

NIS92 CORPORATE NIST, The digital signature standard, Communications of the ACM, v.35 n.7, p.36-40, July 1992

Pang92 James Y.C. Pang, Deepinder S. Gill, and Songnian Zhou. Implementation and Performance of Cluster-Based File Replication ~n Large-Scale Distributed Systems. Technical report, Computer Science Research Institute, University of Toronto, August 1992.

Rive92 R. Rivest. The MD4 Message-Digest Algorithm. Request for Comments 1320, Network Working Group, ISI, April 1992.

Rost93 E. Rosti , E. Smirni , T. D. Wagner , A. W. Apon , L. W. Dowdy, The KSR1: experimentation and modeling of poststore, Proceedings of the 1993 ACM SIGMETRICS conference on Measurement and modeling of computer systems, p.74-85, May 10-14, 1993, Santa Clara, California, United States

Sand85 Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon. Design and implementation of the Sun Network Filesystem. In Proc. of the Summer 1985 USENIX, pages 119-130, June 1985.

Sand92 Harjinder S. Sandhu , Songnian Zhou, Cluster-based file replication in large-scale distributed systems, Proceedings of the 1992 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, p.91-102, June 01-05, 1992, Newport, Rhode Island, United States

Saty89 M. Satyanarayanan, Integrating security in a large distributed system, ACM Transactions on Computer Systems (TOCS), v.7 n.3, p.247-280, Aug. 1989

Saty90 Mahadev Satyanarayanan, Scalable, Secure, and Highly Available Distributed File Access,

Computer, v.23 n.5, p.9-18, 20-21, May 1990

Spas94 Marjana Spasojevic and M. Satyanarayanan. A Usage Profile and Evaluation of a Wide-Area Distributed File System. In Proc. of the Winter 1994 USENIX, January 1994.

Thom87 James Gordon Thompson. Efficient Analysis of Caching Syatems. PhD thesis, University of California at Berkeley, 1987.

Wolf89 J. Wolf, The placement optimization program: a practical solution to the disk file assignment problem, Proceedings of the 1989 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, p.1-10, May 23-26, 1989, Oakland, California, United States

↑ CITINGS *20*

Tammo Spalink , John H. Hartman , Garth A. Gibson, A Mobile Agent's Effects on File Service, IEEE Concurrency, v.8 n.2, p.62-69, April 2000

Werner Vogels, File system usage in Windows NT 4.0, ACM SIGOPS Operating Systems Review, v.33 n.5, p.93-109, Dec. 1999

Steven D. Gribble , Gurmeet Singh Manku , Drew Roselli , Eric A. Brewer , Timothy J. Gibson , Ethan L. Miller, Self-similarity in file systems, ACM SIGMETRICS Performance Evaluation Review, v.26 n.1, p.141-150, June 1998

Prashant Shenoy , Pawan Goyal , Harrick M. Vin, Architectural considerations for next generation file systems, Proceedings of the seventh ACM international conference on Multimedia (Part 1), p.457-467, October 30-November 05, 1999, Orlando, Florida, United States

Luo Guangchun , Zhang Jun , Lu Xianliang , Lu Jun, HCCM: a novel cache consistence mechanism, ACM SIGOPS Operating Systems Review, v.37 n.2, p.25-36, April 2003

Prashant Shenoy , Pawan Goyal , Harrick M. Vin, Architectural considerations for next-generation file systems, Multimedia Systems, v.8 n.4, p.270-283, July 2002

Keith A. Smith , Margo I. Seltzer, File system aging—increasing the relevance of file system benchmarks, ACM SIGMETRICS Performance Evaluation Review, v.25 n.1, p.203-213, June 1997

Avraham Leff , Joel L. Wolf , Philip S. Yu, Efficient LRU-Based Buffering in a LAN Remote Caching Architecture, IEEE Transactions on Parallel and Distributed Systems, v.7 n.2, p.191-206, February 1996

Windsor W. Hsu , Alan Jay Smith , Honesty C. Young, The automatic improvement of locality in storage systems, ACM Transactions on Computer Systems (TOCS), v.23 n.4, p.424-473, November 2005

Mustaque Ahamad , Rammohan Kordale, Scalable Consistency Protocols for Distributed Services, IEEE Transactions on Parallel and Distributed Systems, v.10 n.9, p.888-903, September 1999

Ian Foster , David Kohr, Jr. , Rakesh Krishnaiyer , Jace Mogill, Remote I/O: fast access to distant storage, Proceedings of the fifth workshop on I/O in parallel and distributed systems, p.14-25, November 17-17, 1997, San Jose, California, United States

Tammo Spalink , John H. Hartman , Garth A. Gibson, A Mobile Agent's Effects on File Service, IEEE

Concurrency, v.8 n.2, p.62-69, April 2000

Vijay Sundaram , Prashant Shenoy, Bandwidth allocation in a self-managing multimedia file server, Proceedings of the ninth ACM international conference on Multimedia, September 30-October 05, 2001, Ottawa, Canada

Yuanyuan Zhou , Zhifeng Chen , Kai Li, Second-Level Buffer Cache Management, IEEE Transactions on Parallel and Distributed Systems, v.15 n.6, p.505-519, June 2004

Jeanna Neefe Matthews , Drew Roselli , Adam M. Costello , Randolph Y. Wang , Thomas E. Anderson, Improving the performance of log-structured file systems with adaptive methods, ACM SIGOPS Operating Systems Review, v.31 n.5, p.238-251, Dec. 1997

Hongzhou Liu , Tom Roeder , Kevin Walsh , Rimon Barr , Emin Gün Sirer, Design and implementation of a single system image operating system for ad hoc networks, Proceedings of the 3rd international conference on Mobile systems, applications, and services, June 06-08, 2005, Seattle, Washington

Je-Ho Park , Vinay Kanitkar , Alex Delis, Logically Clustered Architectures for Networked Databases, Distributed and Parallel Databases, v.10 n.2, p.161-198, September 2001

T. E. Anderson , M. D. Dahlin , J. M. Neefe , D. A. Patterson , D. S. Roselli , R. Y. Wang, Serverless network file systems, ACM SIGOPS Operating Systems Review, v.29 n.5, p.109-126, Dec. 3, 1995

Thomas E. Anderson , Michael D. Dahlin , Jeanna M. Neefe , David A. Patterson , Drew S. Roselli , Randolph Y. Wang, Serverless network file systems, ACM Transactions on Computer Systems (TOCS), v.14 n.1, p.41-79, Feb. 1996

Gang Ma , Adnan Khaleel , A. L. Narasimha Reddy, Performance Evaluation of Storage Systems Based on Network-Attached Disks, IEEE Transactions on Parallel and Distributed Systems, v.11 n.9, p.956-968, September 2000

↑ INDEX TERMS

Primary Classification:

C. Computer Systems Organization

↳ **C.2** COMPUTER-COMMUNICATION NETWORKS

Additional Classification:

C. Computer Systems Organization

↳ **C.2** COMPUTER-COMMUNICATION NETWORKS

↳ **C.4** PERFORMANCE OF SYSTEMS

↳ **Subjects:** Performance attributes; Modeling techniques

General Terms:

Design, Measurement, Performance

↑ Collaborative Colleagues:

Thomas E. Anderson:	Eric J. Anderson	Joseph M. Hellerstein	Margaret Martonosi	Stefan Savage
	Remzi H. Arpaci	Paul Horton	Clifford J. Mather	Stefan R. Savage
	Jean-Loup Baer	Anna R. Karlin	Jeanna Neefe Matthews	James B. Saxe
	Brian N. Bershad			Vivian Shen

	Satish Chandra Adam M. Costello Patrick J. Crowley David E. Culler Michael Dahlin Michael D. Dahlin Andrea C. Dusseau Joel A. Fine James Frew Douglas P. Ghormley Susan L. Graham Steven D. Gribble Anoop Gupta	Randy H. Katz Arvind Krishnamurthy Keith Krueger John D. Kubiatowicz Roger Kumpf James R Larus Edward D. Lazowska Dennis C. Lee Henry M. Levy Kester Li Lok T. Liu Steven Lucco Rich P. Martin Richard P. Martin	Nick McKeown Jeanna M. Neefe Kathleen M. Nichols Michael Olson John K. Ousterhout Susan S. Owicki David Patterson David A. Patterson David Petrou Bradley Richards Steven H. Rodrigues Drew Roselli Drew S. Roselli Drew Shaffer Roselli	Neil Timothy Spring Charles P. Thacker Amin Vahdat Amin M. Vahdat Mohammad Amin Vahdat Robert Wahbe Randolph Wang Randolph Y. Wang David J. Wetherall and the NOW team
Michael D. Dahlin:	Thomas E. Anderson Joel A. Fine James Frew Clifford J. Mather Jeanna M. Neefe Michael Olson David Patterson David A. Patterson Drew S. Roselli Arunkumar Venkataramani	Randolph Y. Wang Praveen Yalagandula		
Clifford J. Mather:	Thomas E. Anderson Michael D. Dahlin David A. Patterson Randolph Y. Wang			
David A. Patterson:	Glenn D. Adams Eric Anderson Thomas E. Anderson Remzi H. Arpaci Andrea C. Arpaci-Dusseau Remzi H. Arpaci-Dusseau Remzi Hussein Arpaci-Dusseau Satoshi Asami Krste Asanovic Walter E. Baker James Beck Bidyut K Bose Bidyut Kumar Bose Aaron Brown Aaron B. Brown Aaron Baeten Brown Denice L. Champlin Peter M Chen Peter M. Chen Ann L. Chervenak Ann Chervenak-	Daniel T. Fitzpatrick John K. Foderaro James Frew Richard J. Gallagher Phil Garrison Garth Gibson Garth A. Gibson Paul M Hansen Paul M. Hansen Paul Mark Hansen John H. Hartman Yong Q He Yong Qiang He Joseph M. Hellerstein Lisa Hellerstein John Hennessy John L. Hennessy Daniel Hettena Paul N. Hilfinger Mark Hill Mark D Hill	John Kubiatowicz Jon Kuroda Howard A. Landman James R Larus Edward D. Lazowska Corinna G Lee Daebum Lee Edward K Lee Edward K. Lee Eva K. Lee Karl Lew Calvin Ling Mark A. Linton Dimitris Lioupis Lok T. Liu K. Lutz Ken Lutz Gurmeet Singh Manku David Martin Clifford J. Mather Robert N. Mayo	Roger C. Raphael Scott A Ritchie Drew S. Roselli Carlo H. Séquin Abhijit Sahay Eunice Santos Klaus E Schauser Martin E. Schulze Carlo H. Sequin S. Seshan Srinivasan Seshan Robert W. Sherburne Robert W. Sherburne Ken Shirriff Tim Sippel Michael Stonebraker Ramesh Subramonian Nisha Talagala Madhusudhan Talluri

	Drapeau Leonard Chung David E. Culler Michael D. Dahlin Michael Donald Dahlin Peter J. Denning Alvin M. Despain David R. Ditzel Ann L. Drapeau Ann L. C Drapeau Andrea C. Dusseau Korbin Van Dyke Susan J. Eggers Eric Evers E. Scott Fehr Joel Fine Joel A. Fine	Mark D. Hill D. A. Hodges Ryan Huebsch David Judd William Kakes Richard M. Karp Manolis G. H. Katevenis Manolis G.H. Katevenis Randy H. Katz Kimberly Keeton Kimberly Kristine Keeton John Keller Shing Kong Shing I. Kong Christoforos Kozyrakis Christoforos E. Kozyrakis	Ethan Miller Ethan L. Miller Marguerite Murphy Jeanna M. Neefe Chris Nyberg Michael Olson David Oppenheimer David Lawrence Oppenheimer John K. Ousterhout James B. Peek J. M. Pendleton Zvi Peshkess Richard S. Piepho Mukul R. Prasad Lisa Pullen Ranae Ralson	George S. Taylor Randi Thomas Noah Treuhaft Richard Tuck Amin M. Vahdat Korbin S. Van Dyke Randolph Y. Wang David A. Wood David A. Wood Katherine Yelick Katherine A. Yelick Robert Yung Benjamin G. Zorn and the NOW team Thorsten von Eicken
Randolph Y. Wang:	Thomas E. Anderson Adam M. Costello David E. Culler Michael D. Dahlin Nitin Garg Arvind Krishnamurthy Junwen Lai Kai Li Rich P. Martin Richard P. Martin	Clifford J. Mather Jeanna Neefe Matthews Jeanna M. Neefe David Patterson David A. Patterson Drew Roselli Drew S. Roselli Klaus E. Schauser Chris J. Scheiman Sumeet Sobti	Katherine Yelick Xiang Yu Chi Zhang Fengzhou Zheng Elisha Ziskind	

↑ **Peer to Peer - Readers of this Article have also read:**

- Data structures for quadtree approximation and compression **Communications of the ACM** 28, 9
  Hanan Samet

- A hierarchical single-key-lock access control using the Chinese remainder theorem **Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing**
  Kim S. Lee , Huizhu Lu , D. D. Fisher

- Putting innovation to work: adoption strategies for multimedia communication systems **Communications of the ACM** 34, 12
  Ellen Francik , Susan Ehrlich Rudman , Donna Cooper , Stephen Levine

- The GemStone object database management system **Communications of the ACM** 34, 10
  Paul Butterworth , Allen Otis , Jacob Stein

- An intelligent component database for behavioral synthesis **Proceedings of the 27th ACM/IEEE conference on Design automation**
  Gwo-Dong Chen , Daniel D. Gajski

↑ **This Article has also been published in:**

- **ACM SIGMETRICS Performance Evaluation Review**
  Volume 22 , Issue 1 May 1994

Local networks

Full text Pdf (3.01 MB)

Source **ACM Computing Surveys (CSUR)** archive
Volume 16 , Issue 1 (March 1984) table of contents
Pages: 3 - 41
Year of Publication: 1984
ISSN:0360-0300

Author William Stallings Honeywell Information Systems, Inc., McLean, VA

Publisher ACM Press New York, NY, USA

↑ ABSTRACT

The rapidly evolving field of local network technology has produced a steady stream of local network products in recent years. The IEEE 802 standards that are now taking shape, because of their complexity, do little to narrow the range of alternative technical approaches and at the same time encourage more vendors into the field. The purpose of this paper is to present a systematic, organized overview of the alternative architectures for and design approaches to local networks.

The key elements that determine the cost and performance of a local network are its topology, transmission medium, and medium access control protocol. Transmission media include twisted pair, baseband and broadband coaxial cable, and optical fiber. Topologies include bus, tree, and ring. Medium access control protocols include CSMA/CD, token bus, token ring, register insertion, and slotted ring. Each of these areas is examined in detail, comparisons are drawn between competing technologies, and the current status of standards is reported.

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 ALLAN, R. 1982. Local-area networks spur moves to standardize data communications among computers and peripherals. Electron. Des. Dec. 23, 107-112.

2 ALLAN, R. 1983. Local networks: Fiber optics gains momentum. Electron. Des. June 23.

3 ANDREWS, D. W., AND SCHULTZ, C_v. D. 1982. A token-ring architecture for local area

networks: An update, in Proceedings of COMPCON Fall 82 (Washington, D.C., Sept. 20-23). IEEE Computer Society, Los Angeles, pp. 615-624.

4 ANSI 1982. Dra{t Proposed American National Standard Local Distributed Data Interface. American National Standards Institute, New York.

5 E. Arthurs , B. W. Stuck, A theoretical performance analysis of polling and carrier sense collision detection communication systems, Proceedings of the seventh symposium on Data communications, p.156-162, October 27-29, 1981, Mexico City, Mexico

6 BOSEN, R. 1981. A low-speed local net for under $100 per station. Data Commun. 10, 2 (Dec.), 81- 83.

7 William E. Burr, An overview of the proposed american national standard for local distributed data interfaces, Communications of the ACM, v.26 n.8, p.554-561, Aug. 1983

8 Bux, W. 1981. Local-area subnetworks: A performance comparison. IEEE Trans. Commun. COM-29, 10 (Oct.).

9 Bux, W., CLOSS, F., JANSON, P. A., KUMMERLE, K., MILLER, H. R., AND ROTHAUSER, H. 1982. A local-area communication network based on a reliable token ring system. In Proceedings of the International Symposium on Local Computer Networks.

10 CHRISTENSEN, G. S. 1979. Links between computerroom networks. Telecommunications 13, 2 (Feb.), 47-50.

11 CLANCY, G. J., et al. 1982. The IEEE 802 Committee states its case concerning its local network standards efforts. Data Commun. 11, 4 (Apr.), 13, 238.

12 COOPER, E. 1982. 13 often-asked questions about broadband. Data Commun 11, 4 (Apr.), 137-142.

13 COOPER, E. 1983. Broadband network design: Issues and answers. Comput. Des. 22, 3 (Mar.), 209-216.

14 COOPER, E., AND EDHOLM, P. 1983. Design issues in broadband local networks. Data Commun. 12, 2 (Feb.), 109-122.

15 DEC 1980. The Ethernet: A Local Area Network Data Link Layer and Physical Layer Spec~ficatwns, Sept. 30. Digital Equipment Corp., Intil Corp., and Xerox Corp., Digital Equipment Corp., Maynard, Mass.

16 DERFLER, F., AND STALLINGS, W. 1983. A Manager's Guide to Local Networks. Prentice-Hall, New York.

17 DINESON, M. A., AND PICAZO, J. J. 1980. Broadband technology magnifies local network capability. Data Commun. 9, 2 (Feb.), 61-79.

18 DIXON, R. C. 1982. Ring network topology for local data communications. In Proceedings o { COMP- CON, Fall 82 (Washington, D.C., Sept. 20-23). IEEE Computer Society, Los Angeles, pp. 591- 605.

19 W. D. Farmer , E. E. Newhall, An experimental distributed switching system to handle bursty computer traffic, Proceedings of the first ACM symposium on Problems in the optimization of data communications systems, p.1-3, October 13-16, 1969, Pine Mountain, Georgia, United States

20 FORBES, J. 1981. RF prescribed for many local links. Data Commun. 10, 9 (Sept.).

21 FRANTA, W. R., AND CHLAMTEC, I. 1981. Local Networks. Lexington Books, Lexington, Mass.

22 FREEDMAN, D. 1983. Fiber optics shine in local area networks. Mini-Mwro Syst. 16, 10 (Sept.), 225- 230.

23 GORDON, R. L., FARR, W. W., AND LEVINE, P. 1980. Ringnet: A packet switched local network with decentralized control. Comput. Networks 3, 373-379.

24 GRAUBE, M. 1982. Local area nets: A pair of standards. IEEE Spectrum (June), 60-64.

25 HAFNER, E. R., NENADAL, Z., AND TSCHANZ, M. 1974. A digital loop communications system. IEEE Trans. Commun. COM-22, 6 (June), 877- 881.

26 HAHN, M., ANO BELANGER, P. 1981. Network minimizes overhead of small computers. Electronics, Aug. 25.

27 HEYWOOO, P. 1981. The Cambridge ring is still making the rounds. Data Commun. 10, 7 (July), 32- 36.

28 HOHN, W. C, 1980. The Control Data loosely coupled network lower level protocols. In Proceedings o{ the National Computer Con{erence (Anaheim, Calif., May 19-22), vol. 49. AFIPS Press, Reston, Va., 129-134.

29 HOPKINS, G. T. 1979. Multimode communications on the MITRENET. Proceedings of the Local Area Communwations Network Symposium (Boston, May). M~tre Corp., McLean, Va., pp. {69- 178.

30 HOPKINS, G. T., AND MEISNER, N. B. 1982. Choosing between broadband and baseband }local networks. Mini-Micro Syst. 16, 7 (June).

31 HOPPER, A. 1977. Data ring at Computer Laboratory, University of Cambridge. In Local Area Networking. NBS Publ. National Bureau of Standards, Washington, D.C., pp. 500-531, 11-16.

32 HUBER, D., STEINLIN, W., AND WILD, P. 1983. SILK: An implementation of a buffer insertion ring. IEEE J. Selected Areas Commun. SAC-l, 5 (Nov.), 766-774.

33 IBM CORP. 1982. IBM Series/1 Local Communicatwns Controller Feature Description. GA34-0142- 2. IBM Corporation.

34 IEEE 1983. IEEE Project 802, Local Network Standards. Institute of Electrical and Electronic Engineers, New York.

35 KRUTSCH, T. E. 1981. A user speaks out: Broadband or baseband for local nets? Data Commun. 10, 12 (Dec.), 105-112.

36 Liu, M. T. 1978. Distributed loop computer networks. In Advances in Computers, vol. 17. Academic Press, New York, pp. 163-221.

37 LIU, M. T., HILAL, W., AND GROOMES, B. H. 1982. Performance evaluation of channel access protocols for local computer networks. In Proceedings o{ COMPCON FALL 82 (Washint,~on, D.C., Sept. 20-23). IEEE Computer Society, Los Angeles, pp. 417-426.

38 LUCZAK, E. C. 1978. Global bus computer communication techniques. In Proceedings of the

Symposium on Computer Networks (Gaitherstmrg, Md., Dec.). IEEE Computer Society, Los Angeles, pp. 58-67.

39 MALONE, J. 1981. The microcomputer connection to local networks. Data Commun. 10,12 (Dec.), 101- 104.

40 MARKOV, J. D., ANO STROLE, N. C. 1982. Tokenring local area networks: A perspective. In Procee&ngs o{ COMPCON FALL 82 (Washington, D.C., Sept. 20-23). IEEE Computer Society, Los Angeles, pp. 606-614.

41 Robert M. Metcalfe , David R. Boggs, Ethernet: distributed packet switching for local computer networks, Communications of the ACM, v.19 n.7, p.395-404, July 1976

42 METCALFE, R. M., BOGGS, D. R., THACKER, C. P., ANO LAMPSON, B. W. 1977. Multipoint data communication system with collision detection. U.S. Patent 4,063,220.

43 MILLER, C. K., ANO THOMPSON, D. M. 1982. Making a case for token passing in local networks. Data Commun. 11, 3 (Mar.), 79-88.

44 MYERS, W. 1982. Towards a local network standard. IEEE Micro 2, 3 (Aug.), 28-45.

45 NELSON, J. 1983. 802: A progress report. Datamation (Sept.), 136-152.

46 PARKER, R. 1983. Committees push to standardize disk I/O. Comput Des 22, 3 (Mar.), 30-34.

47 PENNY, B. K., ANO BAGHOADI, A. A. 1979. Survey of computer communications loop networks. Comput. Commun. 2, 4 (Aug.), 165-180; 2, 4 (Oct.), 224-241.

48 PIERCE, J. R. 1972. Network for block sw~tches of data. Bell Syst. Tech. j 51, 6 (July-Aug.).

49 RAVCH-HINoIN, W. 1982. IBM's local network scheme. Data Commun. 11, 5 (May), 65-70.

50 RAWSON, E., ANO METCALFE, R. 1978. Fibernet: Multimode optical fibers for local computer networks. IEEE Trans. Commun. COM-26, 7 (July), 983-990.

51 ROMAN, G. S. 1977. The design of broadband coaxial cable networks for mult~mode communications. MITRE Tech. Rep. MTR-3527. Mitre Corp., McLean, Va.

52 ROSENTHAL, R., Ed. 1982. The selection of local area computer networks. NBS Special Publ. 500-96, National Bureau of Standards, Washington, D.C., Nov.

53 Jerome H. Saltzer , David D. Clark , Kenneth T. Pogran, Why a ring?, Proceedings of the seventh symposium on Data communications, p.211-217, October 27-29, 1981, Mexico City, Mexico

54 SALTZER, J. H., AND POGRAN, K. T. 1979. A starshaped ring network with high maintainability. In Proceedings of the Symposium on Local Area Communications Network (Boston, May). Mitre Corp., McLean, Va., pp. 179-190.

55 SALWEN, H. 1983. In praise of ring architecture for local area networks. Comput. Des. 22, 3 (Mar.), 183-192.

56 SHOCH, J. F. 1980. An Annotated Bibliography on Local Computer Networks. Xerox Palo Alto Research Center, Palo Alto, Calif., Apr.

57 SHOCH, J. F., DALA, Y. K., ANO REDELL, D. D. 1982. Evolution of the Ethernet local computer network. Computer 15, 8 (Aug.), pp. 1-27.

58 STAHLMAN, M. 1982. Inside Wang's local net architecture. Data Commun. 11, 1 (jan.), 85-90.

59 William Stallings, Local Networks, Macmillan Library Reference, 1990

60 William Stallings, Local Networks, Prentice Hall PTR, Upper Saddle River, NJ, 1984

61 STIEGLITZ, M. 1981. Local network access tradeoffs. Comput. Des 20, 12 (Oct.), 163-168.

62 STUCK, B. 1983a. Which local net bus access is most sensitive to congestion? Data Commun. 12, 1 (Jan.), 107-120.

63 S?UCK, B. 1983b. Calculating the maximum mean data rate in local area networks. Computer 16, 5 (May), 72-76.

64 THORNTON, J. E. 1980. Back-end network approaches. Computer 13, 2 (Feb.), 10-17.

65 Carl Tropper, Local Computer Network Technologies, Academic Press, Inc., Orlando, FL, 1981

66 WILKES, M. V., AND WHEELER, D. J. 1979. The Cambridge digital communication ring. In Proceedings of the Symposium on Local Area Communwations Network (Boston, May). Mitre Corp., McLean, Va., pp. 47-62.

67 YEN, C., AN.D CRAWFORO, R. 1983. Distribution and equalization of signal on coaxial cables used in 10 Mbits baseband local area networks. IEEE Trans. Commun. COM-31, 10 (Oct.), 1181-1186.

↑ CITINGS *3*

L. Press, Benchmarks for LAN performance evaluation, Communications of the ACM, v.31 n.8, p.1014-1017, Aug. 1988

Yasushi Kiyoki , Kazuhiko Kato , Takashi Masuda, A stream-oriented approach to distributed query processing in a local area network, Proceedings of the 1986 ACM SIGSMALL/PC symposium on Small systems, p.146-155, December 1986, San Francisco, California, United States

Bandula W. Abeysundara , Ahmed E. Kamal, High-speed local area networks and their performance: a survey, ACM Computing Surveys (CSUR), v.23 n.2, p.221-264, June 1991

↑ INDEX TERMS

Classification:

B. Hardware

↳ **B.4** INPUT/OUTPUT AND DATA COMMUNICATIONS

↳ **B.4.1** Data Communications Devices

↳ **Subjects:** Transmitters**; Receivers (e.g., voice, data, image)

C. Computer Systems Organization

↳ **C.2** COMPUTER-COMMUNICATION NETWORKS

General Terms:
Design, Performance, Standardization

↑ **REVIEW**

"Roland Norman Ibbett"

This paper is a concerned principally with the hardware technology of local networks. After a general introduction to the subject of local networks (note that local area networks are introduced later as a subset), Section 1 of the paper surveys the topologies and tansmission media that are currently appropriate for local networks and considers the relationship between them. The author observes that the choices of transmission medium and topology cannot be made independently, although some of the restrictions suggested are based on assumptions about available equipment rather than fundamental technological limitations. It is assumed, for instance, that the central switching element in a star topology network is a (relatively slow) circuit switch, and thus requires nothing etter than a twisted pair for its implementation. High-speed packet switched stars are quite feasible, however, and indeed a multiplicity of each switches can be interconnected by coaxial cable or optical fibre to form a tree (such networks have been constructed at the University of Manchester in the UK (Centrenet) and by American Bell [1]. In this paper the tree topology is assumed to be nothing more than the general case of a bus, with extended spurs as branches. Three types of network are defined: Local Area Networks (LANs), High-Speed Local Networks (HSLNs) and computerized branch (i.e., telephone) exchanges. The remainder of the paper concentrates on the first two of these types, with Section 2 of the pape presenting a very thorough survey and comparison of LAN and HSLN networking techniques based on the bus topology. *Online Computing Reviews Service*

↑ **Peer to Peer - Readers of this Article have also read:**

- The effect of latency on user performance in Warcraft III **Proceedings of the 2nd workshop on Network and system support for games**
  Nathan Sheldon , Eric Girard , Seth Borg , Mark Claypool , Emmanuel Agu

- Learning subjective relevance to facilitate information access **Proceedings of the fourth international conference on Information and knowledge management**
  James R. Chen , Nathalie Mathé

- Data structures for quadtree approximation and compression **Communications of the ACM** 28, 9
  Hanan Samet

- Learning and the reflective journal in computer science **Australian Computer Science Communications** 24, 1
  Susan E. George

- A hierarchical single-key-lock access control using the Chinese remainder theorem **Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing**
  Kim S. Lee , Huizhu Lu , D. D. Fisher

A Quantitative Analysis of Cache Policies for Scalable Network File Systems

Michael D. Dahlin, Clifford J. Mather, Randolph Y. Wang,
Thomas E. Anderson, and David A. Patterson

Computer Science Division, University of California at Berkeley
{dahlin, cjmather, rywang, tea, pattrsn}@cs.berkeley.edu

Abstract

Current network file system protocols rely heavily on a central server to coordinate file activity among client workstations. This central server can become a bottleneck that limits scalability for environments with large numbers of clients. In central server systems such as NFS and AFS, all client writes, cache misses, and coherence messages are handled by the server. To keep up with this workload, expensive server machines are needed, configured with high-performance CPUs, memory systems, and I/O channels. Since the server stores all data, it must be physically capable of connecting to many disks. This reliance on a central server also makes current systems inappropriate for wide area network use where the network bandwidth to the server may be limited.

In this paper, we investigate the quantitative performance effect of moving as many of the server responsibilities as possible to client workstations to reduce the need for high-performance server machines. We have devised a cache protocol in which all data reside on clients and all data transfers proceed directly from client to client. The server is used only to coordinate these data transfers. This protocol is being incorporated as part of our experimental file system, xFS. We present results from a trace-driven simulation study of the protocol using traces from a 237 client NFS installation. We find that the xFS protocol reduces server load by more than a factor of six compared to AFS without significantly affecting response time or file availability.

1 Introduction

Current network file systems rely on powerful central servers that make it difficult to build economical large-scale file systems. Ideally, a network file system should scale to hundreds or thousands of client machines using nothing more than commodity workstations, even for the server. In reality, the widely used SUN Network File System, NFS [Sand85], has spawned a new industry dedicated to building the high-performance multiprocessor systems needed to scale NFS to more than a few dozen clients. The Andrew File System, AFS [Howa88], was designed to reduce server load relative to NFS in the interest of scalability, but its ultimate scalability is limited because AFS still relies on a central server to receive a copy of all modified data and to supply data for all client cache miss requests. Also, NFS and AFS must generally use specialized servers rather than commodity desktop workstations as server machines because the server must support enough disks to hold a copy of the entire file system, and desktop workstations are generally limited to a single SCSI string. Commodity workstations are a more cost effective way to buy computing power and I/O bandwidth because server machines must be designed with greater I/O expandability and because development costs of the more complicated servers must be amortized over a smaller sales volume. For instance in the SUN product line, a server costs three times as much as a similarly configured workstation.

Trends in file system use promise to place even heavier demands on the central servers of file systems. Baker et al. [Bake91] report that the size of large files grew by an order of magnitude between 1985 and 1992. If this trend continues, the cost of transferring all data through the central server may become prohibitive. File systems are also being asked to manage data over wide area networks (WANs) where bandwidth restrictions limit the amount of data that can be supplied from a central source.

At the same time, technology trends are giving clients tremendous amounts of disk space, main memory, and processing power and are also providing high-speed low-latency networks to tie these resources together. Inexpensive disks make it feasible for clients to store large amounts of data locally. A 1.3 GB SCSI disk currently costs less than $1000, and most workstations sold today are configured with significant amounts of local disk. Similarly, the aggregate memories and processing resources of client machines dwarf the capacity of a single server machine. Additionally, high speed local area networks (LANs) allow clients to access data from peers across a local area network almost as quickly as they can access local data.

This paper investigates the quantitative benefits of utilizing cache techniques oriented towards extreme scalability to reduce the load on the central server. These techniques provide better and more cost-effective file service than a specialized server machine by pushing responsibilities onto the clients in the system to exploit the aggregate client disk, processing, and memory capacities. The protocol has four pieces inspired by efforts to achieve scalable cache coherence in massively parallel processors [Arch86, Leno90]. The protocol uses a no write through policy, utilizes client-to-client data transfers, implements write ownership, and takes advantage of cluster servers.

In this paper we compare the effects of this protocol to a baseline AFS system. We base this comparison on event driven simulation parameterized to model the service demands of file system requests on a DECstation 5000. We compared the performance of the systems under a workload taken from a large NFS system at

This work is supported in part by the Advanced Research Projects Agency (N00600-93-C-2481), the National Science Foundation (CDA 8722788), California MICRO, Digital Equipment Corporation, the AT&T Foundation, Xerox Corporation, and Siemens Corporation. Dahlin was also supported under a National Science Foundation Graduate Research Fellowship. Anderson was also supported by a National Science Foundation Young Investigator Award.

SIGMETRICS 94- 5/94 Santa Clara, CA. USA
© 1994 ACM 0-89791-659-x/94/0005..$3.50

Berkeley in which an Auspex file server provides service for 237 client workstations[1].

Our principal result is that for this workload the experimental protocol reduced server load by more than a factor of six. In addition, we show that each of the four parts of the protocol has a significant impact on performance, that the protocol not only reduces average server load but also significantly reduces peak demand at the server, that the aggregate client memories are more effective at reducing disk I/O than the server memory, and that cluster servers isolate almost all communication to within clusters, as is desirable when clusters are connected by a WAN.

This study also addresses a number of issues that arise when clients become responsible for more file system services. We examine the problems of scalable backup, data availability in the presence of client failures, and security when clients supply data to each other.

We are currently implementing the protocol described here as part of a file system called xFS. To facilitate comparison with AFS, this paper assumes the AFS policy of synchronizing file consistency when a file is closed for writing and assumes whole-file caching. However our implementation of xFS does not have these restrictions. The actual xFS implementation also stripes data in a RAID distributed across client disks to improve I/O bandwidth and availability. We do not consider these other issues further.

Section 2 of this paper outlines the file system caching algorithms used by NFS and AFS and motivates the alternative strategies we consider in this paper. Section 3 describes our workload, and in Section 4 we discuss the key aspects of our simulation. Section 5 details the results of our study, with emphasis on the impact of the protocol on server scalability, network load, and client load. Section 6 examines the potentially thorny issues of backup, availability, and security that arise when clients are given responsibilities that were formerly the server's. We survey related network file system studies in Section 7. Finally in Section 8 we summarize our conclusions.

2 File System Cache Protocols

An important factor in a file system's scalability is its caching policy. File systems use caches to improve response time and to reduce server load. Clients can access data found in their memory caches more quickly than they can access remote data on a server. File caches reduce server load by satisfying some requests without server interaction. The use of caches, however, introduces the problem of *cache consistency*: different caches may hold copies of the same file, and if the file is changed by one client, the changes must be seen when the file is read by a different client. How the cached copies are kept consistent can have a large effect on server scalability. This section describes the cache protocols found in the industry-standard NFS, the emerging AFS standard, and our more scalable xFS.

2.1 Existing Protocols

NFS, the current industry-standard distributed file system protocol, was designed to provide good response time for moderate numbers of clients rather than to provide scalability. NFS caches file system data in main memory on each client workstation. NFS does not attempt to use the client's local disk space as a cache, nor does it attempt to keep file data strictly coherent. Instead, periodic invalidations of file attribute information ensure that new data eventually (within several seconds) replace any out-of-date cached copies. Once a file's attributes are invalidated, the next time the file is referenced the client will verify that its cached copy is current. If it is not, the client will fetch the new data from the server. Clients write through all modified file data to the server to ensure that fetches from other clients will receive the new data. NFS maintains separate caches for data, attributes, and names, and the protocol caches data on a per-block basis.

NFS's scalability is limited by its use of relatively small in-memory file caches rather than the larger caches possible if local disks were used. NFS's policy of periodically invalidating attributes guarantees a stream of client requests to the server as attributes expire, even for files that are not being modified. NFS's write through policy sends all changes to the server disk, even if no other clients are using the data.

AFS improves upon NFS's scalability by using a large local on-disk cache at each client and by using callbacks for cache consistency. AFS uses a two-level cache on each client. An in-memory file cache similar to NFS's file cache provides good response time for most accesses, but misses to the in-memory file cache go to a client on-disk file cache and only go to the server if not satisfied there. Rather than requiring periodic verification of a file's consistency as NFS does, AFS reduces server load further by using *callbacks*: the server maintains a list of all cached copies of each data file and notifies clients when another client modifies the file. The client fetches the new data from the server the next time it opens the file. AFS clients send all modified data to the server when a file is closed, guaranteeing that the server has the most current version of the data and allowing the server to know when to invalidate the other cached copies. Clients also cache directory information in write through directory caches. All modifications to directories are sent immediately to the server, which maintains callbacks to keep cached copies of directory information consistent.

Despite these improvements, AFS's scalability is still limited. All communication and data transfer takes place between the clients and the server; no direct client-to-client communication is allowed. In particular, the server supplies data each time a client has a cache miss and receives data each time a client closes a file it has written. The central server must have enough disk space to store all of the file system data; this is despite the fact that the aggregate size of the client disks is typically much larger than that of the server disks. The server is also responsible for fielding all directory modification operations and for generating callback messages on every cache coherence operation.

2.2 xFS Protocol

In this paper we consider the effect of four separate optimizations to the AFS protocol. Together, these push most server responsibilities onto the client machines. Collectively we refer to these as the xFS protocol. Each of these optimizations has been proposed as a way to improve the scalability of multiprocessor hardware caches, and some have also been suggested for file systems. In Section 5.2 we evaluate their performance impact individually and find that all are important to get good performance.

The first two optimizations, write through and client-to-client data transfers, eliminate file transfers through the server, making the server responsible only for coordinating the data that flow from client to client. These two aspects of the protocol also eliminate the

[1] The Auspex is built with special hardware to allow it to support this large number of clients [Hitz90].

need to buy a specialized server machine configured with a large amount of disk space.

1. **No write through.** Clients no longer write modified data to the server on close. Instead they inform the server of the update, and the server invalidates cached copies at other clients using callbacks. Modified files remain on each client's local disk.

The elimination of write through is motivated by a number of studies showing that when a client writes a file, it is often deleted or quickly rewritten by the same client [Thom87, Bake91, Blaz91, Kist92]. We confirmed this pattern for the Berkeley NFS traces. After discarding the statistics for writes made during the last day of the simulation, we found that of the bytes that would be sent to the server in AFS, 85% were overwritten or deleted without being read by another client, another 5% were never read by another client, and only 10% were read by another client. With no write through, bytes that are overwritten can be discarded at the client without ever being transferred to the server. Note that there is no need to write data to the server to ensure the data's durability. Delayed writes of about 30 seconds have been used in other network file systems to reduce writes to the server without putting too much data at risk of loss in a client crash [Nels88], but xFS's use of client disks allows a complete no write through policy where files are never written to the server.

2. **Client-to-client data transfers.** When a client has a cache miss, it sends a request for the data to the server, and the server forwards the request to a client that is currently caching the needed data. The second client then sends the desired data directly to the client that wants the data.

Client-to-client data transfers reduce server load by replacing a large server data transfer with a small forwarding packet. Direct client-to-client communication also permits the no write through policy to be implemented without the significant delays that would be incurred if all requested dirty data were first written to the central server and then supplied by the server. Client-to-client data transfers are an example of separating the control and data paths as suggested by the Mass Storage Reference Model [Coyn93].

The first two parts of the protocol allow all data to be stored on client disks, implementing what is referred to for multiprocessors as a cache only memory architecture (COMA) [Hage92, Rost93]. An important detail of this approach is that we must guarantee that the clients don't discard the last copy of any file. The clients do this by marking one copy of each file as permanent. A copy becomes marked when it is written, and marked copies may be passed between clients but not discarded. When a client's cache is full it sends any marked copies it would normally discard to a randomly selected client. The client notifies the server of this transfer. These marked data copies are otherwise managed in the same way as unmarked data copies.

The final two optimizations, write ownership and clustering, try to reduce the demands on the server of coordinating cached copies of files.

3. **Write ownership based cache consistency.** The first time a client closes a modified file the server is notified, triggering an invalidation of all copies cached on other clients. At that point the client has exclusive *write ownership* [Arch86] and may modify the file freely without notifying the server; there are no other copies of the data to be invalidated. A client will lose exclusive ownership of a file when another client opens the file for reading. Its copy will be invalidated if another client acquires exclusive ownership.

Write ownership is an optimization of the write invalidate consistency protocol on which AFS's callback mechanism is based. It allows us to eliminate messages to the server in the common case of repeated writes by the same client to the same file.

4. **Clustering.** Clusters are formed by selecting groups of workstations that closely cooperate or are near each other on the network topology, for instance, on the same LAN. Cluster servers keep track of the ownership and callback state for all of the clients in the cluster. The central server only tracks file location information to the cluster level, relying on the cluster server to forward requests to the specific clients caching data. The cluster servers isolate ownership changes and data transfers internal to the cluster from the central server. For instance, if ownership is transferred between two clients in the same cluster, the cluster server is notified of the change, but the central server need not be. On the other hand, if ownership is transferred between clients from different clusters, the central server must be involved so it can know that a new cluster server is responsible for tracking the ownership of the file.

Clustering is inspired by the DASH multiprocessor architecture [Leno90] which clusters processing nodes on busses as we cluster clients on LANs. Clustering improves scalability by offloading some central server state to cluster servers and isolating the central server from changes in state only affecting clients in the cluster [Blaz91, Munt92, Sand92]. Clustering also allows the system to work in a wide area network context by organizing communication around the cluster LAN networks and using the WAN links only when necessary.

xFS clustering is distinct from name space splitting and read replication, two methods of utilizing multiple servers available to NFS and AFS. Name space splitting improves file system scalability by manually splitting the file system into logical pieces, each managed by a different server. However, it can be difficult to divide files among servers so as to balance load and avoid hot spots [Wolf89]. Name space splitting is, however, useful when the different parts of the file system are managed by different administrative domains. xFS can support this splitting by using multiple "central" servers, with each cluster server providing a combined file system view to its clients. File systems can also be replicated across multiple servers to improve scalability for reading files, at a cost of making file writes more expensive [Lisk91, Kist92]. We will show in Section 5 that xFS-style clustering reduces the cost of both reads and writes.

The combination of these four changes allows xFS to be dramatically more scalable than AFS. The central server is no longer involved in any data transfers and coordinates a much smaller amount of control activity: the central server must forward read miss requests between clusters when they cannot be satisfied within a cluster; the central server must send consistency messages between clusters when a modification in one cluster invalidates cached copies in another; finally, the central server is informed when files are created or deleted so that it always knows what files exist and where to forward requests for all files.

3 Trace Overview

To evaluate the performance impact of these changes we gathered traces of NFS file system activity from a large NFS installation served by an Auspex file server. The system includes 237 clients spread over four Ethernets, each of which connects directly to the central server. The trace spans seven days, and unless noted, the measurements that appear in this paper cover the last six days of the trace after using the first day's activity to warm the caches. During the full seven day trace 141,574 files were referenced.

We gathered this trace by monitoring network activity on each of the four Ethernets. On each subnet we placed a workstation that monitored all network traffic using rpcspy [Blaz93] which is built on the Ultrix Packetfilter interface [Mogu87]. Over the trace period, rpcspy reported that it dropped 4% of all network traffic calls due to buffer overflow.

We postprocessed the NFS trace to reflect the semantics of the AFS and xFS protocols. Since we gathered the traces at the network, we had access only to NFS network traffic, which introduced some biases of NFS into our raw trace. For instance NFS has no network-visible open or close calls, and many `getattr` (get file attribute) calls are really for validating cache consistency.

In the first step of the postprocessing we added opens and closes to the trace. We added file opens before the first access to an unopened file. Read and read/write opens signal AFS and xFS to bring the file being accessed into the local on-disk cache. We inserted file closes immediately after the last file access before a long (2 minute) period of inactivity for the file or before a block write to block zero of a file after a series of writes to other parts of the file. We use AFS's write close semantics: after the close, the newly written file should be supplied to any subsequent read open.

Block reads and writes in the trace are caused by NFS in-memory cache misses. In AFS and in the version of xFS simulated here, these reads and writes cause local disk traffic, but no network activity, since whole file caching is assumed and file consistency is handled when the file is opened or closed.

We included NFS directory reads and writes as AFS and xFS directory reads and writes. Directories were simulated with the semantics that each directory write is immediately visible to the entire system. AFS implements this by writing directory changes through to the server while xFS uses its file ownership and invalidation mechanisms.

Finally, we include most NFS `getattr` calls as simulator requests for file attributes. We excluded `getattr` calls immediately before an access to a block of the same file, assuming those calls to be NFS cache validation packets. The simulator also dynamically eliminates many `getattr` calls by filtering calls through an attribute cache. The attribute cache is kept consistent in the same way as the directory cache for each protocol. An attribute is invalidated when the file it references is written. Note that we are not simulating the "access time" attribute, which is updated for each file read, for either AFS or xFS.

The resulting trace is similar to other measured AFS workloads in macro characteristics. Our simulated AFS server supplied on average 5.0 MB to each of its clients per day for read opens; [Spas94] measured 5.3 MB per client per day for a large AFS installation. We measured a 5.7 MB per client per day write back load; 4.7 MB per client per day loads were measured by [Spas94].

This trace reflects the file system activity of a real system. Although this enhances our confidence that the trace is realistic, the capabilities of the traced system can limit the activity seen in the trace. The prime example of this limitation is on peak load. Our trace will underestimate the peak server load that might be imposed on a more scalable system for two reasons. First, the limited speed of the traced system will spread out requests, resulting in longer periods of activity but lower peaks. Second, users will tend to avoid operations that take a long time on the traced system, lowering both peak and overall load. Sharing is another example where the system's limitations may distort the workload. Since NFS has weak data sharing semantics, few users attempt to share data. If more files were shared, both AFS and xFS would see increased server loads, although AFS's increase would be larger since AFS's sharing requires data transfer through the server while sharing under xFS is accomplished with read forwarding and invalidation packets.

4 Simulator Methodology

We built a simulator to evaluate the performance of AFS and xFS for the traced workload. This simulator starts with a model of system behavior describing what actions are taken to implement the AFS and xFS protocols. We then parameterized the system to reflect the performance of real hardware. The subsections below describe the system model and the hardware parameters used.

4.1 System Model

Our simulator provides both average resource utilization and more detailed performance information. In the simplest case we can determine average processor, disk, and network utilizations by simulating cache behavior on the trace input and counting accesses to the different hardware resources. We get more detailed performance information by adding an event driven model to the cache simulation to measure the response time of different requests and monitor the burstiness of the utilization of different parts of the system. This event-driven hardware model includes both hardware and queuing delays. The rest of this subsection provides details about the simulated caches.

Our simulations of xFS and AFS include both on-disk and in-memory client file caches, and our AFS simulations include an in-memory file cache at the server. These caches are simulated using whole-file caching for simplicity, although in practice both AFS and xFS would cache chunks of files. We do break large transfers into 64 KB chunks for realistic latency measurements. We assume in-memory caches of 8 MB per client and 128 MB at the AFS server, and we give each client a 100 MB on-disk file cache.

Our simulations also include attribute caches used when clients access a file's attributes without fetching the entire file. Each client had a 2048 entry in-memory attribute cache backed by its disk, and the server has a 32,768 entry in-memory cache. The server supplies attributes and the systems maintain attribute consistency using the same protocols used for the files themselves.

Because cache behavior is so crucial to performance of large scale file systems, we warm the caches before gathering statistics. The results presented in this paper are gathered during the last six days of our seven day trace, after warming the caches for the first day, a Saturday. Figure 1 plots the hit rate of read opens not satisfied completely in the in-memory cache over time and indicates that after the first day, the hit rate fluctuates between 30% and 95%. There appears to be no general upward trend as we would expect once the caches are warm. The steady state hit rate is rela-

tively low because opens that are completely satisfied in the NFS local in-memory cache did not appear in the trace

Even after warming the caches, 8% of the read opens (21% of those that miss on the local disk) access files that have not been referenced earlier in the trace We must make some assumption about which xFS client owns these files. We arbitrarily assume each file with unknown location is stored on a randomly selected client disk. The impact of this assumption is limited since 92% of the file opens are located normally, at a client that is currently caching the data.

4.2 Hardware Parameters

To estimate the performance of xFS and compare it to AFS, absent an implementation, we parameterized the model to reflect the performance of a mid-range workstation for tasks similar to those that would be performed in an xFS or AFS implementation. We approximate the performance of a DECstation 5000/200 using measured and reported performance results for its subsystems summarized in Figure 2. Each hardware resource services a *requestSize* request in time *overhead+requestSize/bandwidth*. This approach is clearly an oversimplification: not all requests to a given piece of hardware will have the same overheads and bandwidths and the actual overheads are unlikely to exactly match those for current systems. Nevertheless, these simple assumptions provide a starting point for system evaluation. [Lazo86] used a similar approach in parameterizing performance for network file system simulations.

The processor overhead time represents the CPU and memory subsystem time to send or receive one network request and do a small amount of work in the file system. We estimated this time by measuring the time for a DECstation 5000/200 to handle an NFS `getattr` request. For the CPU bandwidth for large requests, we use the time to supply file system data from the in-memory file cache reported in [Chen93].

We assume that the machines use disks that rotate at 5400 RPM, that the typical seek time is 4ms[2], and that the disk bandwidth is 2MB/s.

We base the network topology on the configuration of the clients in our NFS trace: four subnets each connected to the server. For AFS, each subnet connects directly to the server, and for xFS each subnet connects to a cluster server. The cluster servers connect to the central server using a fifth subnet. The network latency is the time to transmit a minimum-sized Ethernet packet and the network bandwidth is an optimistic estimate of the net bandwidth available on an Ethernet

While other performance assumptions would result in differences in the absolute latency and burstiness numbers reported later, they are unlikely to affect our central conclusion that the xFS protocol scales significantly better than AFS.

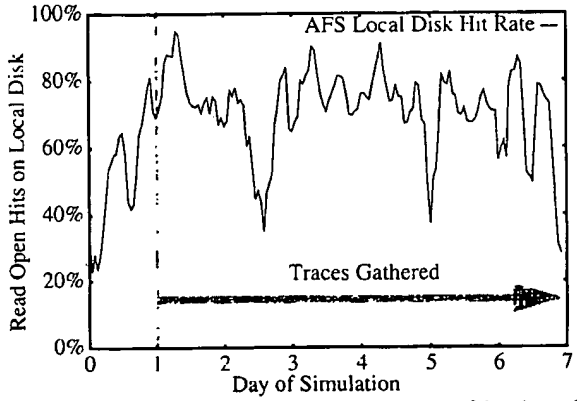


Figure 1. Local hit rate over time. This plot of local on-disk cache hit rate against day of the simulation of AFS suggests that one day is sufficient to warm the caches. Hour-to-hour fluctuations have been smoothed in this plot by averaging over the previous 4 hours for each point.

	Overhead	Bandwidth
Processor	1.4 ms	7 MB/s
Disk	9.6 ms	2 MB/s
Network	0.1 ms	4 or 5 x 1 MB/s

Figure 2. Service demand parameters.

5 Results

This section presents the results of our simulations. We show that the proposed optimizations reduce server load by more than a factor of six compared to AFS, and we also show that the xFS protocol greatly reduces peak bursts of server load. xFS also significantly reduces total network load, and the distribution of traffic that remains is better suited for a mixed LAN/WAN environment. The increased responsibility this protocol places on clients does increase client file system load slightly, but the extra forwarding of read requests does not increase response time.

The next subsection presents our overall results in more detail, and the subsection after that examines the individual impact of each of the four main aspects of xFS: its no write through policy, client-to-client transfers, write ownership, and clustering. We find that all four techniques make significant contributions to the overall performance.

5.1 Overall Results

This section compares the xFS protocol to AFS in terms of server CPU load, the burstiness of server load, response time, network load, and client load.

Figure 3 summarizes our results showing that xFS reduces server load by more than a factor of six compared to AFS. This load estimate is the total server processor demand including both overhead and bandwidth as described in Section 4.2, expressed as a fraction of AFS's server demand. We find that xFS reduces server load by 85% compared to AFS by eliminating data transferred at the server and by reducing the number of messages the server must handle.

	Server Messages	Server Data	Server Load
AFS	1,411,504	15.2 GB	1.000
xFS	457,356	0.0 GB	0.153

Figure 3. Total server load. The normalized server load expresses the server CPU load for the simulated protocols as a fraction of the simulated server load for AFS.

[2] This estimate of the typical disk overhead differs from the "average" seek time reported by manufacturers because it accounts for locality seen in real workloads [Henn90] while the manufacturer-reported average seek is the mean time over all possible source and destination tracks—seeks that average one third of the distance across the disk surface

Figure 4 details how much server load each type of operation demands. Write close operations at the server include write through (for AFS), notifying the server of a write of a file that is not write owned (for xFS), and the messages sent by the server to invalidate cached copies. Read open operations at the server are caused by client misses and include handling the client request and supplying the data (for AFS) or forwarding the request (for xFS). Delete operations include the message sent to the server to indicate that a file has been deleted and the server messages notifying the clients caching that file. Attribute messages include packets to request, update, and invalidate file attribute information. The category "other" includes all other packets sent to or received by the server; an xFS client notifies the server when it purges a file (potentially forwarding marked data to another cache) to make room for new data.

In addition to the total work at the server, performance and scalability will also be dependant on periods of heavy load. Figure 5 summarizes the distribution of time spent at increasing levels of server load for AFS and xFS. It shows that xFS's reduction in average load translates into a reduction in time spent at high load. This figure indicates that the AFS server spends several minutes per day working at loads of over 0.5 while the xFS server is never loaded that heavily. The extremely low peak demands of xFS suggest that we could scale the system to a larger number of clients than AFS. We note that the absolute load level for both machines is relatively low, suggesting that either server could probably handle the Berkeley Auspex workload. As we noted earlier, however, the maximum load that either system experiences in this simulation is limited by the maximum load accepted by the NFS system where the workload trace was gathered.

Having servers forward read requests can potentially increase latency. Our measurements show, however, that the aggregate effect of the client in-memory caches minimizes the impact of the extra step. We focus on the time spent to open a file for reading, from when the request is issued until the first chunk of up to 64 KB arrives on the local disk. We consider both requests that are found on the local disk without additional network communication and requests that are satisfied over the network.

Figure 6 breaks down the response time based on where requested files are found. Opens that are satisfied locally, requiring no disk accesses, account for most of the opens and are satisfied quickly by both systems. Misses that are satisfied in the remote in-memory cache and misses that require remote disk accesses are slightly slower in the xFS implementation because of the extra forwarding step. This does not, however, increase the cost of a miss because xFS misses are satisfied by the remote client in-memory cache more often than AFS requests are satisfied by the server in-memory cache. The higher remote client in-memory cache hit rate is initially surprising because the AFS server in-memory cache is 128 MB while each xFS client cache is just 8 MB. We note, however, that the aggregate size of the 237 client caches is 1896 MB, making them together an effective file cache even though many of the files stored in this distributed cache are duplicates. Further, note that the server does not attempt to keep track of which clients have a file cached in memory rather than on disk; although this would be an obvious optimization, it could increase server load. Figure 7 plots the remote in-memory hit rate for xFS as a function of client memory cache size and indicates that the 128 MB server cache is about equivalent to 5 MB client caches.

	AFS	xFS
Write Close	0.429	0.018
Read Open	0.453	0.073
Delete	0.014	0.013
Attribute	0.104	0.043
Other	0.000	0.006
Total	**1.000**	**0.153**

Figure 4. Server load breakdown. Portion of AFS server load due to each type of request.

Network load, the number of bytes transferred over the network during the trace, is an important metric of scalability. We are particularly concerned about minimizing network usage for wide area network file systems, where network bandwidth can become a

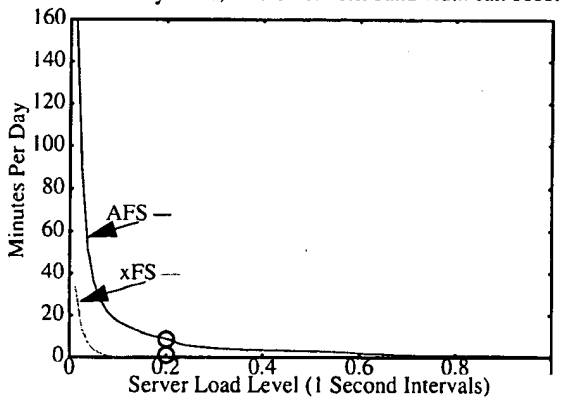


Figure 5. Cumulative distribution of server load. The X axis is the amount of load presented to the server during a one second interval. This load is the sum of the service demands for all requests that arrive at the server during a one second interval. The Y value is the amount of time during the day that the server experienced at least that load level. The AFS server handled at least one request per second for 159 minutes per day, while the xFS server was completely idle for all but 34 minutes per day. The circled points indicate that the AFS server would have a load of more than 0.2 for over eight minutes per day while the xFS server would have a load that high less than ten seconds per day.

	AFS		xFS	
	Freq.	**Time**	**Freq.**	**Time**
Local	60%	6.3 ms	60%	6.3 ms
Remote	40%	57 ms	40%	56 ms
Mem	9%	25 ms	15%	30 ms
Disk	31%	66 ms	25%	71 ms
Total	**100%**	**27 ms**	**100%**	**26 ms**

Figure 6. Read open response time. The response time is the time needed to put the first chunk of the opened file onto the local disk and return. Local hits are data that are already on the local disk. For both AFS and xFS data not found in the local on-disk cache are fetched from a remote machine. For AFS that remote machine is the server but for xFS that remote machine is another client. At that remote machine the desired data may be found on disk or in the in-memory cache.

bottleneck. Bandwidth can also be an issue for mobile computing using wireless interconnects [Kist92].

Figure 8 summarizes total network traffic for AFS and xFS using the assumption that each packet sent has a header of 128 bytes. The table indicates that xFS reduces total network traffic by 52%. The major difference in total network bandwidth is xFS's elimination of write through traffic for files that are later modified or deleted by the same client.

Although xFS only reduces total network traffic by a factor of two compared to AFS, it significantly changes the nature of that traffic. Figure 9 shows that client-to-client transfers reduce the bytes transferred to the server by more than 99%. This reduction is crucial for file systems where the server may be located across a WAN. The use of clustering also reduces the number of bytes transferred out of the cluster to less than 20% of AFS's total traffic, a consideration when clusters are separated by gateways or WANs.

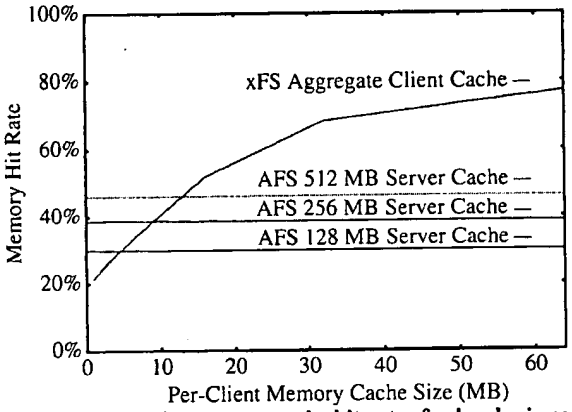


Figure 7. Remote in-memory cache hit rates for local misses.

	AFS	xFS
Packets	1,411,504	1,968,242
Overhead	180 MB	252 MB
Data Bytes	15,251 MB	7,222 MB
Write Through	8,096 MB	0 MB
Other Data	7,154 MB	7,222 MB
Total Bytes	15,431 MB	7,474 MB

Figure 8. Network traffic for xFS and AFS from all sources. The total bytes transferred is an estimate formed by adding the total data bytes transferred plus 128 bytes per request to reflect protocol overhead and control information. The packet count for xFS reported here differs from the number of messages reported in Figure 3 because Figure 3 only considered traffic to and from the server.

Finally, we note that overall client load is increased only slightly even though clients shoulder considerably more responsibility in xFS than in AFS, for instance by supplying data to other clients. The xFS protocol increases the total amount of file system work done by clients by 10% for the measured workload. The increase in client load is small because most of a client's load comes from local block reads and writes which are unchanged in xFS. Further, although xFS increases the load on each client to handle data requests from other clients, the reduced write through activity largely offsets this increase. This fraction would be smaller still if the trace included the even larger amount of file system activity that is purely local, such as reads that hit the client's in-memory cache. Figure 10 shows that the demands on clients are not greatly altered by the xFS protocol.

	AFS	xFS
Central Server	15,431 MB	58 MB
Other Out Cluster	N/A	2,902 MB
In Cluster	N/A	4,514 MB

Figure 9. Total network traffic over different parts of the network. The total includes both data and a 128 byte per-packet header. In a WAN or large-scale environment the connection to the central server or to other clusters may be slower than the network within a cluster.

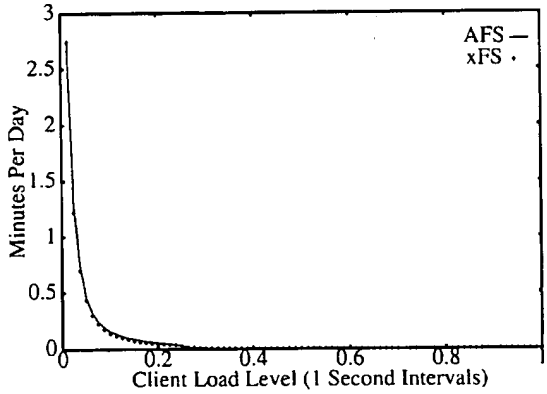


Figure 10. Cumulative distribution of one-second load levels at clients. The average client processor is active doing file system activity for less than three minutes per day. The xFS and AFS client loads are almost indistinguishable.

Protocol	Write	Read	Delete	Attr.	Other	Total
AFS	0.429	0.453	0.014	0.104	0.000	1.000
+ no write through	***0.113***	***0.496***	0.014	0.102	***0.003***	***0.728***
+ client-to-client	0.113	***0.175***	0.014	0.102	***0.015***	***0.418***
+ write ownership	***0.032***	0.175	0.014	0.102	0.015	***0.337***
+ clusters (full xFS)	***0.018***	***0.073***	***0.013***	***0.043***	***0.006***	***0.153***

Figure 11. Server load by type of activity and protocol. The AFS line indicates the server load stemming from write closes, read opens, deletes, attribute operations, and other operations. Each subsequent line shows the breakdown and total after one more part of the xFS protocol is added. Server loads that changed significantly from the previous line are highlighted.

5.2 Protocol Breakdown

In the previous section, we considered the aggregate effect of all four optimizations studied. Here we consider their individual effects. We conclude that each of the optimizations contributes significantly to the performance of the xFS protocol. Figure 11 summarizes the load as each part of the xFS protocol is added to the system. This section explains the benefits of each of the strategies in more detail

Simply eliminating write through from AFS would reduce the server load by 27% for this workload. The server load associated with closing files that have been written would be reduced by nearly a factor of four because clients only need to send a small notification message to the server rather than transmitting the modified file in one or more larger messages. However, the server load associated with supplying read misses is increased slightly as the server endures write backs of modified files that other clients want to read. 10% of the bytes that were written to the server by AFS are later read and show up as increased read load. Another small load, in the category *Other*, comes from write throughs that must eventually be made to free cache space.

Utilizing client-to-client data transfers reduces the server load by an amount equal to 0.31 times the original AFS load. This reduction comes from the elimination of data transfers through the server on read opens. Note, however, that the work in the category *Other* is increased slightly. This increase is from messages clients send when they free space by discarding files from their caches. The server must be informed when even clean files are discarded so that it doesn't forward read requests to a client no longer caching the desired data.

Write ownership reduces the number of messages processed by the server, and therefore server load, by an additional 25% compared to the client-to-client line. Figure 12 indicates that over 80% of the messages notifying the server that a file has been closed are eliminated using write ownership.

Cluster servers reduce messages of all types by intercepting requests that would have been handled by the central server. Cluster servers reduce the server load for reading files and reading attributes by forwarding requests that can be satisfied within the cluster. This read forwarding is the primary benefit of cluster servers. The number of write close messages is also reduced, and this reduction comes from two sources that combine to reduce server load by 0.014 times AFS's load. First, a few write closes transfer ownership between clients of the same cluster. These writes are handled by the cluster server, reducing central server load by 0.003 times AFS's original load. More significantly, the cluster server acts to fan out invalidation packets from the central server. One invalidation packet from the central server to the cluster server is sufficient to invalidate all data copies in the cluster. These invalidations reduce the central server load by 0.011 times the original load. The load of delete messages and *Other* messages processed by the central server is also reduced slightly. Some delete messages, invalidating multiple copies of the same file in caches in one cluster, are eliminated because the cluster server distributes these messages to the appropriate clients in the cluster. *Other* messages are reduced when a file discarded from a cache is still cached in some client in the cluster; in that case the server may still forward requests for that file to the cluster and so need not be notified of the change. Finally, note that the four cluster servers' loads ranged from 0.05 to 0.20 as a fraction of the AFS server load. In other words, they have about the same load as the xFS central server.

Protocol	Notify Server	Invalidate Client
no write through	291,198	46,015
+ownership	50,423	46,015

Figure 12. Write close messages without and with write ownership. *Notify Server* messages tell the server to invalidate any other cached copies of a file. It invalidates files with *Invalidate Client* messages.

We also considered extending the strategy of write ownership of files to include ownership of directory sub-trees to allow us to avoid notifying the server of all file creations and deletions. If a client owned a directory, it would notify the server of file creations and deletions when ownership of the directory containing the file was lost. This strategy would exploit the common case of files being created, used, and deleted without ever being seen by another client. Our simulations did not measure the benefits of directory ownership, but they allow us to place an upper bound on the benefits by noting that of the 0.018 load for write closes, 0.009 was for newly created files and of the 0.013 load for deletes, 0.012 was for messages notifying the server of the delete. If all of these messages could be omitted, the total server load would be reduced by 0.021 units, a 14% reduction from the xFS protocol in this unrealistically optimistic case. We conclude that this improvement would not justify the considerable added complexity of the approach. If some of the other sources of load were reduced further, this reduction would be a more significant fraction of the remaining load, and this decision would have to be reexamined.

6 Challenges for Decentralized Operation

Although the xFS protocol's reliance on client disk caches improves scalability, it introduces three potential challenges for reliable operation. We must provide backup that scales with the number of clients; we must ensure that the files are highly available despite being distributed over many disks; and we must provide security guarantees so that unauthorized clients cannot read or change data they store. We find that the data replication that is a natural part of the xFS protocol makes backup easier and increases availability. Also, message digests can be used to provide security for data supplied by other clients.

6.1 Backup

xFS's ability to manage multiple data copies in normal operation can be used to manage the backup copies of the data as well. This simplifies the design of the system and also allows us to use multiple backup archives to scale the backup bandwidth as the rest of the system scales.

xFS treats each archive as another client, and the server keeps track of the backup copies of data just as it tracks other cached copies. A client cache backs up a file by sending it to an archive and telling the server about the new copy. The system must have policies for the frequency of backup, for deciding which of the potentially many clients caching a file is responsible for backing it up (this decision can be made without additional communication), and for retrieving data from the backing store.

We plan to use tertiary storage robots to manage the backup media. Tertiary storage robots provide from hundreds of gigabytes

to tens of terabytes of storage with file access times measured in tens of seconds [Katz91]. The robots provide deep storage as traditional tape systems do, but they have the added advantage that all files are on-line in the sense that a user may access the data without human intervention. Tertiary robots are not a requirement of xFS; backup could be done using traditional off-line tapes. The advantage of using storage robots for backup is that data may be sent to or retrieved from the tertiary storage system without operator intervention, allowing the system to automatically provide services such as access to old versions of files or deleted files.

Note that the server's disk need not be backed up: the server can reconstruct its list of cached copies and metadata by polling the cluster servers [Nels88].

Backup over the network exerts a small additional load on the system. Although we did not include this load in the simulations, its impact on performance should be small. As noted in Section 2.2, almost all files are overwritten or deleted quickly and so need not be copied from the client to the backup archive. Further, backup may be scheduled for periods of low system load to avoid disturbing regular system activity.

6.2 Availability

xFS's second challenge is availability. As the file system is spread over more machines, the probability that one of the machines containing file system data is unavailable increases. Availability problems are mitigated by large client caches, file replication, and the file access patterns observed in our trace. Even higher availability could be achieved using explicit data replication.

Large client caches and file replication from caching and backup reduce xFS's vulnerability to unavailable clients. Large client caches provide some insulation—the crash of one machine will often not be noticed by others [Kist92]. xFS also automatically stores redundant copies of shared read files in different caches increasing the availability of those files, and on-line backup provides added copies of older files. These properties of xFS mean that only a few files, those recently written but not backed up and not read by a second client, are vulnerable to single point failures. Since files written by one client are seldom read by another, these vulnerable files are seldom accessed when their writer is down.

We estimate from our trace of file activity that the average client will go hundreds of days without noticing file unavailability stemming from the crash of another client. This low rate suggests that the xFS protocol will not significantly change data availability which will still be dominated by the availability of the server. For this calculation we assumed that clients fail randomly with an exponentially distributed mean time to failure of 30 days and an exponentially distributed mean time to repair of one hour. We also assumed that data was backed up to a reliable on-line tape robot every morning at 2 AM. Under these assumptions we found that each day an average of 0.56 of the 237 clients in the trace would try to access data that was cached only on an unavailable client. This figure was based on 500,000 seven day trials and has a 95% confidence interval of ±0.04. Availability could be more of a problem if write-sharing of data were more widespread than seen in our trace. Also, if a user's machine crashes, the user may not be able to switch to an alternate machine to do work since modified data will unavailable until the crashed machine recovers

If stronger availability guarantees are needed, client-to-client data replication of recently modified data provides a scalable solution. Shortly after a client closes a file for writing, it would send the data to one or more other clients [Lisk91, Birr93]. This solution is scalable since it adds no additional server messages if the server knows ahead of time which clients mirror writes to each other. The copy delay chosen is a trade-off between performance and availability guarantees with longer delays significantly reducing client-to-client bandwidth [Bake91] while increasing the length of time the file is vulnerable to a single point failure. In the future we plan to investigate these trade-offs for client-to-client transfers and also plan to look at using striping to reduce the cost of high availability. Our current simulations do not make additional client copies.

6.3 Security

xFS's use of client disks to store and supply data raises two security concerns, data confidentiality and integrity. We do not want clients to transfer data into a cache that is not authorized to read the data, and we do not want to accept altered data from a malicious client on a client-to-client transfer.

We believe that in many environments most clients will trust at least the other clients in the same cluster to enforce the system's data access rules. Communication between trusting clients does not require the steps described here.

The confidentiality of data cached on client disks is also addressed by AFS [Saty89] and the techniques used there apply to xFS as well, for data read by the client. xFS, however, adds one new way that data can be brought into a client's cache: clients flush data to one another as their caches fill. Since this flushing is rare, the performance impact of the chosen strategy will be limited. In the extreme case, if no other clients are trusted with the data, the data could be encrypted before it is flushed. The same client would have to decrypt it if it were later accessed. More commonly, data will only be flushed to a limited subset of trusted clients, for instance only to clients in the same cluster and administrative domain.

xFS can guarantee data integrity, allowing clients to accept data from even untrusted clients, by guaranteeing two things: that a secure copy of each file always exists and that a client can detect when a file has been altered from the secure image.

To guarantee the existence of pristine data, the system must trust the client that created the data—which it must do in any event since it has given the client permission to modify the data—and must trust the on-line backup archive. The client that created the data pins a copy in its cache until the file is backed up to the robotic storage. After the file is backed up, the writer is free to flush the data, since if another client modifies it without permission, the system may still recover the file from the tape robot.

A client verifies untrusted data using a *message digest*, a special checksum that can be calculated efficiently, but for which it is computationally infeasible to create different data to match [Rive92]. The server stores a 128-bit digest for each 64 KB data chunk with its list of chunks cached at the clients. When it forwards a client's request for data, it includes the digest for the pristine data in the forwarding packet. The digest in the forwarding packet is protected using an encrypted digital signature [NIS92]. The protected part of the forwarding packet would also include a request identifier to protect against playback attacks. The client supplying data forwards the protected digest along with the data. The original client then verifies the data supplied against the original digest. If the file is corrupt, the client asks the server for a copy from another source.

Because the work of computing digests is done at the clients, digests do not severely impact server scalability. Digests can be supplied to clients reading data using no additional network packets and they are updated at the server only when file write ownership is lost. When the server forwards a data read request, it must encrypt a short message including the digest and request identifier and append that message to the forwarding packet. If the work of encrypting this message is small compared to sending the packet, digests will not increase server load for read requests. (If encryption is hard compared to sending a message, the unprotected digest may be sent directly to the client requesting the read in a separate message.) Our simulation assumes that the cluster servers are trusted by the clients in the cluster, so once a cluster server knows the digest for a particular file chunk, the cluster server may forward the digest to the appropriate client. Digests only change when a file is written, so clients only calculate a new digest and send it to the server when they lose write ownership.

We simulated message digests assuming that the signature encryption is cheap compared to sending a message. In that case the only additional server load is receiving 15,510 digest updates when write ownership is lost. This increases server load by less than 1% of AFS's total server load.

Message digests do not severely impact response time. We measured the bandwidth to compute the MD4 digest on a DEC Alpha AXP 3000/400 to be 13.3 MB/s. To be consistent with the other processor speeds used in this paper, we simulated MD4 calculations using our measured DECstation 5000/200 MD4 bandwidth of 2.5 MB/s. Even if clients calculate a message digest on all data received, trusting no other clients, the read open time for files is 28 ms, just 2 ms slower than the 26 ms read open time reported in Section 5 for xFS without message digests. Since the MD4 calculation bandwidth exceeds the bandwidth of the network, the impact to performance is minimal. This approach will become even more attractive if processor speed improvements continue to outpace I/O system improvements.

7 Related Work

This paper evaluates the effectiveness of a file system that combines the strategies of eliminating write through, client-to-client transfers, write ownership, and clustering. The fusion of these strategies has produced a system that we believe will scale in size and across wide area networks. This section surveys some other combinations of these schemes that have been suggested as methods to achieve scalable file systems.

The Andrew file system, AFS, was designed with scalability as a main criteria [Howa88, Saty90]. Andrew based scalability on the use of, first, large on-disk client caches to reduce file reads from the server, and second, callbacks to reduce the number of protocol messages handled by the server. xFS is also based on large on-disk client caches and callbacks but generalizes their use using four additional techniques.

The mass storage system reference model [Coyn93] decouples location and name service from the actual storage of data. The model defines a name server and location server that locate the storage server that actually manages the bitfile. Goldick et al. [Gold93] have implemented an AFS-based storage system which allows data to reside in up to 32 separate locations. In xFS each client acts as a storage server, and server and cluster servers together act as a two-level location server. This study indicated that this division greatly reduced the load on the central resource, the central server.

Sprite [Nels88] uses delayed writes to the server to reduce server load. The diskless Sprite clients, however, must write data through to the server within about 30 seconds to reduce vulnerability to crashes. xFS's extends delayed writes to a no write through policy by using the clients' local disks.

Blaze and Alonso [Blaz91, Blaz92] suggest dynamically building hierarchies for widely shared data. Once a server has supplied a threshold number of copies of a file, the server will refuse to supply the data to any more clients. Instead, the request will be forwarded to a client already caching the file. Clients acting as intermediate servers are also responsible for keeping callback information on the files they have supplied to other caches. The authors also suggest a number of strategies which clients may use to guess which other client has desired data without going to the server. These hinting techniques could be applied to an xFS implementation.

Muntz and Honeyman [Munt92] studied the effect of putting an intermediate data server between the central server and the clients in an Andrew system. They found that the hit rates at the intermediate server were surprisingly low. Client caches of 40 MB, small for an on-disk cache, reduced the intermediate cache hit rate to under 20% for both traces studied. The reason is that it is difficult to give the intermediate server a big enough cache to hold significant amounts of data not found in client caches. Because of this result, xFS is designed with intermediate servers that field only consistency requests; the intermediate servers do not store data. We believe it is feasible to provide enough storage on the intermediate servers to hold all of a cluster's consistency information

The Frolic system [Pang92, Sand92] implements replication of files among cluster servers. When a client accesses data from a remote cluster, Frolic creates a copy of the data in the local cluster. Clients use a different protocol, such as NFS, to access data from the local cluster server. Frolic cluster servers differ from xFS cluster servers in that Frolic cluster servers act as intermediate data caches between the clients and remote servers while xFS's cluster servers merely monitor the location of file copies within the cluster. Frolic's concept of a "locating server" responsible for tracking the current owner of a file is similar to xFS's use of the central server. The authors studied the behavior of shared files using a synthetic workload and found that cluster replication improved performance and server load for shared files unless the "degree of cluster locality" was low; clusters do not perform well if files are read by one cluster and quickly invalidated by another.

8 Conclusions

In this paper, we present and evaluate the xFS caching protocol, designed to improve network file system scalability by taking full advantage of clients' processors, memories, and disks. All files are stored at the clients and all data transfers go directly from client to client. The server is used only to coordinate transfers among the clients. xFS reduces the server load necessary for this coordination by using write ownership and clustering, in most cases allowing clients and cluster servers to avoid interacting with the central server.

We evaluated the performance of xFS using a trace-driven simulation of 237 clients. We found that xFS reduced server load by 85% compared to AFS by eliminating server data transfers and by reducing the number of messages to and from the server by 68%. By moving data storage and data transfer responsibilities to the clients, xFS makes it possible to build a large network file system using only commodity desktop workstations, even for the file server.

Acknowledgments

We would like to thank Matt Blaze for providing us with his rpcspy and nfstrace tools; these formed the basis for our trace processing tools. We would also like to thank John Hartman and the anonymous referees whose comments were very helpful in improving this paper.

References

[Arch86] James Archibald and Jean-Loup Baer. Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model. *ACM Transactions on Computer Systems*, 4:273–298, November 1986.

[Bake91] Mary G. Baker, John H. Hartman, Michael D. Kupfer, Ken W. Shirriff, and John K. Ousterhout. Measurements of a Distributed File System. In *Proc. of the 13th Symposium on Operating Systems Principles*, pages 198–212, October 1991.

[Birr93] Andrew D. Birrell, Andy Hisgen, Chuck Jerian, Timothy Mann, and Garett Swart. The Echo Distributed File System. Technical Report 111, Digital Equipment Corp. Systems Research Center, 1993.

[Blaz91] Matt Blaze and Rafael Alonso. Long-Term Caching Strategies for Very Large Distributed File Systems. In *Proc. of the Summer 1991 USENIX*, pages 3–15, June 1991.

[Blaz92] Matt Blaze and Rafael Alonso. Dynamic Hierarchical Caching in Large-Scale Distributed File Systems. In *Proc. of the 12th International Conf. on Distributed Computing Systems*, pages 521–528, June 1992.

[Blaz93] Matt Blaze. *Caching in Large-Scale Distributed File Systems*. PhD thesis, Princeton University, January 1993.

[Chen93] Peter M. Chen and David A. Patterson. A New Approach to I/O Performance Evaluation–Self-Scaling I/O Benchmarks, Predicted I/O Performance. In *Proc. of 1993 ACM SIGMETRICS*, pages 1–12, May 1993.

[Coyn93] Robert A. Coyne and Harry Hulen. An Introduction to the Mass Storage System Reference Model, Version 5. In *Twelfth IEEE Symposium on Mass Storage Systems*, pages 47–53, April 1993.

[Gold93] Jonathan S. Goldick, Kathy Benninger, Woody Brown, Christopher Kirby, Christopher Maher, Daniel S. Nydick, and Bill Zumach. An AFS-Based Supercomputing Environment. In *Twelfth IEEE Symposium on Mass Storage Systems*, pages 127–132, April 1993.

[Hage92] Erik Hagersten, Anders Landin, and Seif Haridi. DDM–A Cache-Only Memory Architecture. *IEEE Computer*, 25(9):45–54, 1992.

[Henn90] John L. Hennessy and David A. Patterson. *Computer Architecture A Quantitative Approach*. Morgan Kaufmann Publishers, Inc., 1990.

[Hitz90] David Hitz, Guy Harris, James K. Lau, and Allan M. Schwartz. Using UNIX as One Component of a Lightweight Distributed Kernel for Multiprocessor File Servers. In *Proc. of the Winter 1990 USENIX*, pages 285–296, 1990.

[Howa88] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West. Scale and Performance in a Distributed File System. *ACM Transactions on Computer Systems*, 6(1):51–81, February 1988.

[Katz91] Randy H. Katz, Thomas E. Anderson, John K. Ousterhout, and David A. Patterson. Robo-Line Storage: Low Latency High Capacity Storage Systems Over Geographically Distributed Networks. Sequoia 2000 Technical Report 91/3, University of California, September 1991.

[Kist92] James J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems*, 10(1):3–25, February 1992.

[Lazo86] Edward D. Lazowska, John Zahorjan, David R. Cheriton, and Willy Zwaenepoel. File Access Performance of Diskless Workstations. *ACM Transactions on Computer Systems*, 4(3):238–268, August 1986.

[Leno90] D. Lenoski, J. Laudon, K. Gharachorloo, A. Gupta, and J. Hennessy. The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor. In *Proc. of the 17th International Symposium on Computer Architecture*, pages 148–159, May 1990.

[Lisk91] Barbara Liskov, Sanjay Ghemawat, Robert Gruber, Paul Johnson, Liuba Shrira, and Michael Williams. Replication in the Harp File System. In *Proc. of the 13th Symposium on Operating Systems Principles*, pages 226–238, October 1991.

[Mogu87] J. Mogul, R. Rashid, and M. Accetta. The Packet Filter: An Efficient Mechanism for User-Level Network Code. In *Proc. of the 11th ACM Symposium on Operating Systems Principles*, 1987.

[Munt92] D. Muntz and P. Honeyman. Multi-level Caching in Distributed File Systems or Your cache ain't nuthin' but trash. In *Proc. of the Winter 1992 USENIX*, pages 305–313, January 1992.

[Nels88] Michael N. Nelson, Brent B. Welch, and John K. Ousterhout. Caching in the Sprite Network File System. *ACM Transactions on Computer Systems*, 6(1), February 1988.

[NIS92] The Digital Signature Standard Proposed by NIST. *Communications of the ACM*, 35(7):36–40, July 1992.

[Pang92] James Y.C. Pang, Deepinder S. Gill, and Songnian Zhou. Implementation and Performance of Cluster-Based File Replication in Large-Scale Distributed Systems. Technical report, Computer Science Research Institute, University of Toronto, August 1992.

[Rive92] R. Rivest. The MD4 Message-Digest Algorithm. Request for Comments 1320, Network Working Group, ISI, April 1992.

[Rost93] E. Rosti, E. Smirni, T. D. Wagner, A. W. Apon, and L.W. Dowdy. The KSR1: Experimentation and Modeling of Poststore. In *Proc. of 1993 ACM SIGMETRICS*, pages 74–85, 1993.

[Sand85] Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon. Design and Implementation of the Sun Network Filesystem. In *Proc. of the Summer 1985 USENIX*, pages 119–130, June 1985.

[Sand92] Harjinder S. Sandhu and Songnian Zhou. Cluster-Based File Replication in Large-Scale Distributed Systems. In *Proc. of 1992 ACM SIGMETRICS*, pages 91–102, June 1992.

[Saty89] Mahadev Satyanarayanan. Integrating Security in a Large Distributed System. *ACM Transactions on Computer Systems*, pages 247–280, August 1989.

[Saty90] Mahadev Satyanarayanan. Scalable, Secure, and Highly Available Distributed File Access. *IEEE Computer*, pages 9–21, May 1990.

[Spas94] Marjana Spasojevic and M. Satyanarayanan. A Usage Profile and Evaluation of a Wide-Area Distributed File System. In *Proc. of the Winter 1994 USENIX*, January 1994.

[Thom87] James Gordon Thompson. *Efficient Analysis of Caching Systems*. PhD thesis, University of California at Berkeley, 1987.

[Wolf89] Joel Wolf. The Placement Optimization Problem: A Practical Solution to the Disk File Assignment Problem. In *Proc. of 1989 ACM SIGMETRICS*, pages 1–10, May 1989.

Local Networks

WILLIAM STALLINGS

Honeywell Information Systems, Inc., McLean, Virginia 22102

The rapidly evolving field of local network technology has produced a steady stream of local network products in recent years. The IEEE 802 standards that are now taking shape, because of their complexity, do little to narrow the range of alternative technical approaches and at the same time encourage more vendors into the field. The purpose of this paper is to present a systematic, organized overview of the alternative architectures for and design approaches to local networks.

The key elements that determine the cost and performance of a local network are its topology, transmission medium, and medium access control protocol. Transmission media include twisted pair, baseband and broadband coaxial cable, and optical fiber. Topologies include bus, tree, and ring. Medium access control protocols include CSMA/CD, token bus, token ring, register insertion, and slotted ring. Each of these areas is examined in detail, comparisons are drawn between competing technologies, and the current status of standards is reported.

Categories and Subject Descriptors: B.4.1 [**Input/Output and Data Communications**]: Data Communications Devices—*receivers*; *transmitters*; B.4.3 [**Input/Output and Data Communications**]: Interconnections (subsystems); C.2.5 [**Computer-Communication Networks**]: Local Networks

General Terms: Design, Performance, Standardization

INTRODUCTION

Local networks have moved rapidly from the experimental stage to commercial availability. The reasons behind this rapid development can be found in some fundamental trends in the data processing industry. Most important is the continuing decrease in cost, accompanied by an increase in capability, of computer hardware. Today's microprocessors have speeds, instruction sets, and memory capacities comparable to medium-scale minicomputers. This trend has spawned a number of changes in the way information is collected, processed, and used in organizations. There is an increasing use of small, single-function systems, such as word processors and small business computers, and of general-purpose microcomputers, such as personal computers and intelligent terminals. These small, dispersed systems are more accessible to the user, more responsive, and easier to use than large central time-sharing systems.

As the number of systems at a single site—office building, factory, operations center, etc.—increases, there is likely to be a desire to interconnect these systems for a variety of reasons, including

- sharing expensive resources, and
- exchanging data between systems.

Sharing expensive resources, such as bulk storage and line printers, is an important measure for cost containment. Although the cost of data processing hardware has dropped, the cost of such essential

CONTENTS

electromechanical equipment remains high. Even in the case of data that can be uniquely associated with a small system, economies of scale require that most of that data be stored on a larger central facility: The cost per bit for storage on a microcomputer's floppy disk is orders of magnitude higher than that for a large disk or tape.

The ability to exchange data is an equally compelling reason for interconnection. Individual users of computer systems do not work in isolation and will want to retain some of the benefits provided by a central system, including the ability to exchange messages with other users, and the ability to access data and programs from several sources in the preparation of a document or the analysis of data.

A Definition

This requirement for communication, both among multiple computer systems within an organization and between those systems and shared resources, is met by the local network, which is defined as follows:

A local network is a communications network that provides interconnection of a variety of data communicating devices within a small area.

There are three significant elements in this definition. First, a local network is a communications network, not a computer network. In this paper, we deal only with issues relating to the communications network; the software and protocols required for attached computers to function as a network are beyond the scope of the paper.

Second, we broadly interpret the phrase "data communicating devices" to include any device that communicates over a transmission medium, including

- computers,
- terminals,
- peripheral devices,
- sensors (temperature, humidity, security alarm sensors),
- telephones,
- television transmitters and receivers,
- facsimile.

Of course, not all types of local networks are capable of handling all of these devices.

Third, the geographic scope of a local network is small, most commonly confined to a single building. Networks that span several buildings, such as those on a college campus or military base, are also common. A network with a radius of a few tens of kilometers is a borderline case. With the use of appropriate technology, such a system can behave like a local network.

Another element that could included in the definition is that a local network is generally a privately owned rather than a public or commercially available utility. Usually a single organization owns both the network and the attached devices.

Some of the key characteristics of local networks are

- high data rates (0.1–100 megabits per second, Mbps),

- short distances (0.1–50 kilometers),
- low error rate (10^{-8}–10^{-11}).

The first two parameters differentiate local networks from two cousins: long-haul networks and multiprocessor systems.

These distinctions between the local network and its two cousins have an impact on design and operation. Local networks generally experience much fewer data transmission errors and lower communications costs than long-haul networks, and cost–performance trade-offs therefore differ significantly. Also, it is possible to achieve greater integration between the local network and the attached devices, as they are generally owned by the same organization.

A major distinction between local networks and multiprocessors systems is the degree of coupling: Multiprocessor systems are tightly coupled, have shared memory, usually have some central control, and completely integrate the communications function, whereas local networks tend to have the opposite characteristics.

Benefits and Pitfalls

We already have mentioned resource sharing as an important benefit of local networks. This includes not only expensive peripheral devices, but also data.

A local network is also more reliable, available to the user, and able to survive failures. The loss of any one system should have minimal impact, and key systems can be made redundant so that other systems can quickly take up the load after a failure.

One of the most important potential benefits of a local network relates to system evolution. In a nonnetworked installation such as a time-sharing center, all data processing power is in one or a few systems. In order to upgrade hardware, existing applications software must be either converted to new hardware or reprogrammed, with the risk of error in either case. Even adding new applications on the same hardware, or enhancing those that exist, involves the risk of introducing errors and reducing the performance of the entire system. With a local network it is possible to gradually replace applications or systems, avoiding the "all-or-nothing" approach. Another facet of this capability is that old equipment can be left in the system to run a single application if the cost of moving that application to a new machine is not justified.

Finally, a local network provides the potential to connect devices from multiple vendors, which would give the customer greater flexibility and bargaining power.

These represent the most significant benefits of a local network. Alas, there are potential pitfalls as well.

There is a certain amount of loss of control in distributed systems; it is difficult to manage this resource, to enforce standards for software and data, and to control the information available through a network. The prime benefit of networking—distributed systems—also incorporates its prime pitfall.

It is likely that data will be distributed in a local network, or at least that access to data will be possible from multiple sources. This raises the problems of integrity of data (e.g., two users trying to update a database at the same time), security of data, and privacy.

Another pitfall might be referred to as "creeping escalation." With the dispersal of systems and ease of adding computer equipment, it becomes easier for managers of suborganizations to justify equipment procurement for their department. Although each procurement may be individually justifiable, the total may well exceed an organization's requirements.

Finally, as we mentioned, one of the significant benefits of a local network is the potential to connect devices from multiple vendors. However, the local network does not guarantee interoperability, that is, that these devices can be used cooperatively. Two word processors from different vendors can be attached to a local network, but will most likely use different file formats and control characters. Some form of format-conversion software will be required to take a file from one device and edit it on the other.

A further discussion of the applications, benefits, and pitfalls of local networks is contained in Derfler and Stallings [1983].

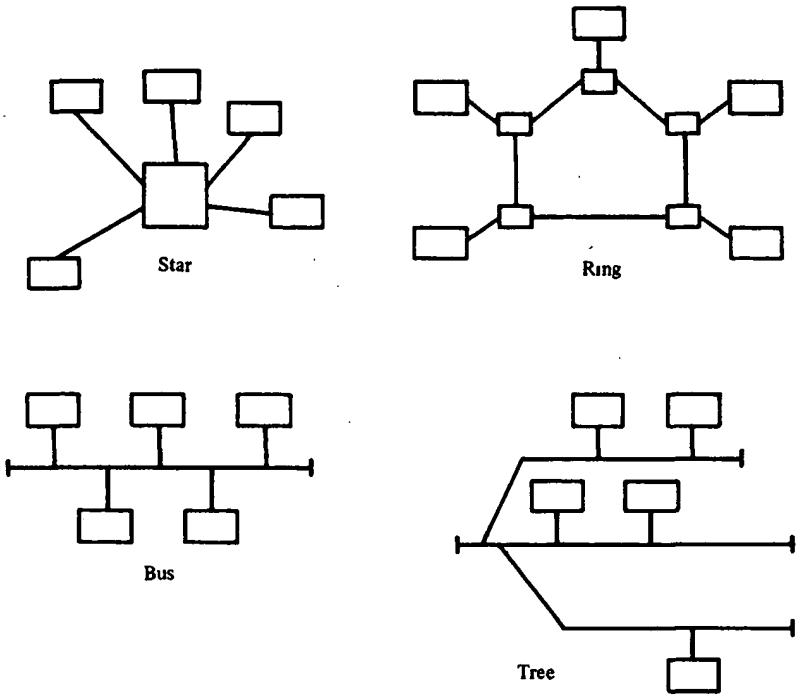


Figure 1. Local network topologies.

1. LOCAL NETWORK TECHNOLOGY

The principle technological alternatives that determine the nature of a local network are its topology and transmission medium [Rosenthal 1982], which determine the type of data that may be transmitted, the speed and efficiency of communications, and even the kinds of applications that a network may support.

In this section we survey the topologies and transmission media that are currently appropriate for local networks, and define three types of local networks based on these technologies.

1.1 Topologies

Local networks are often characterized in terms of their topology, three of which are common (Figure 1): star, ring, and bus or tree (the bus is a special case of the tree with only one trunk and no branches; we shall use the term bus/tree when the distinction is unimportant).

In the star topology, a central switching element is used to connect all the nodes in the network. A station wishing to transmit data sends a request to the central switch for a connection to some destination station, and the central element uses circuit switching to establish a dedicated path between the two stations. Once the circuit is set up, data are exchanged between the two stations as if they were connected by a dedicated point-to-point link.

The ring topology consists of a closed loop, with each node attached to a repeating element. Data circulate around the ring on a series of point-to-point data links between repeaters. A station wishing to transmit waits for its next turn and then sends the data out onto the ring in the form of a packet, which contains both the source and destination address fields as well as data. As the packet circulates, the destination node copies the data into a local buffer. The packet continues to circulate until it returns to the source node, providing a form of acknowledgment.

The bus/tree topology is characterized by the use of a multiple-access, broadcast medium. Because all devices share a common communications medium, only one device can transmit at a time, and as with the

Table 1. Typical Characteristics of Transmission Media for Local Networks

	Signaling technique	Maximum data rate (Mbps)	Maximum range at maximum data rate (kilometers)	Practical number of devices
Twisted pair wire	Digital	1–2	Few	10's
Coaxial cable (50 ohm)	Digital	10	Few	100's
Coaxial cable (75 ohm)	Digital	50	1	10's
	Analog with FDM	20	10's	1000's
	Single-channel analog	50	1	10's
Optical fiber	Analog	10	1	10's

ring, transmission employs a packet containing source and destination address fields and data. Each station monitors the medium and copies packets addressed to itself.

1.2 Transmission Media

Table 1 is a list of characteristics of the transmission media most appropriate for local networks: twisted pair wire, coaxial cable, and optical fiber. The characteristics listed in the table serve to distinguish the performance and applicability of each type of transmission medium.

Twisted pair wiring is one of the most common communications transmission media, and one that is certainly applicable to local networks. Although typically used for low-speed transmission, data rates of up to a few megabits per second can be achieved. One weakness of twisted pair wire is its susceptibility to interference and noise, including cross talk from adjacent wires. These effects can be minimized with proper shielding. Twisted pair wire is relatively low in cost and is usually preinstalled in office buildings. It is the most cost-effective choice for single-building, low-traffic requirements.

Higher performance requirements can best be met by coaxial cable, which provides higher throughput, can support a larger number of devices, and can span greater distances than twisted pair wire. Two transmission methods, baseband and broadband, can be employed on a coaxial cable; these are explained in detail in Sections 2.2 and 2.3, respectively. The key difference is that baseband supports a single data channel, whereas broadband can support multiple simultaneous data channels.

Optical fiber is a promising candidate for future local network installations. It has a higher potential capacity than coaxial cable, and a number of advantages over both coaxial cable and twisted pair wire, including light weight, small diameter, low noise susceptibility, and practically no emissions. However, it has not been widely used thus far due to cost and technical limitations [Allan 1983]. From a technical point of view, point-to-point fiber-optic topologies like the ring will become feasible as the cost of optical fiber drops. Multipoint topologies like the bus/tree are difficult to implement with optical fiber because each tap imposes large power losses and causes optical reflections. One approach to this problem is the passive star coupler (Figure 2) [Freedman 1983; Rawson and Metcalfe 1978]. The passive star coupler consists of a number of optical fibers fused together: Light entering from one fiber on the input side is equally divided among and output through all fibers on the other side of the coupler.

1.3 Relationship between Medium and Topology

The choices of transmission medium and topology cannot be made independently. Table 2 illustrates desirable combinations.

The bus topology can be implemented with either twisted pair wire or coaxial cable.

The tree topology can be employed with broadband coaxial cable. As we shall see, the unidirectional nature of broadband signaling allows the construction of a tree architecture. The bidirectional nature of baseband signaling on either twisted pair wire or coaxial cable would not be suited to the tree topology.

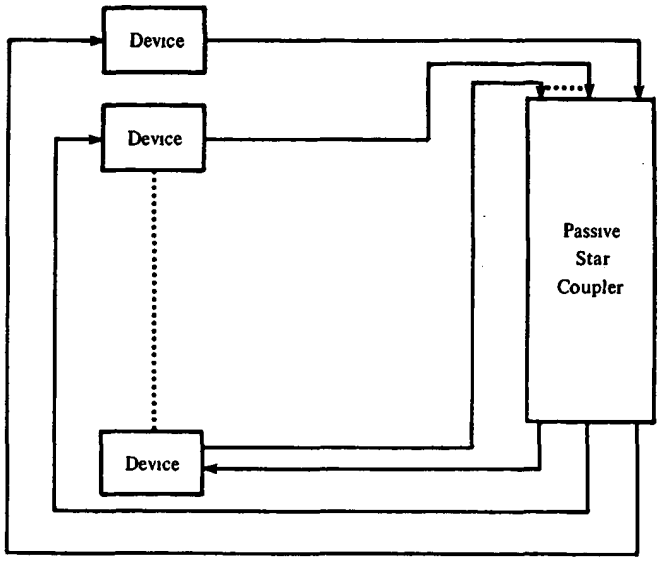


Figure 2. Optical fiber passive star configuration.

Table 2. Relationship between Transmission Medium and Topology

Medium	Topology			
	Bus	Tree	Ring	Star
Twisted pair	×		×	×
Baseband coaxial	×		×	
Broadband coaxial	×	×		
Optical fiber			×	

The ring topology requires point-to-point links between repeaters. Twisted pair wire, baseband coaxial cable, and fiber all can be used to provide these links. Broadband coaxial cable would not work well in this topology, as each repeater would have to be capable, asynchronously, of receiving and retransmitting data on multiple channels. It is doubtful that the expense of such devices could be justified.

The star topology requires a single point-to-point link between each device and the central switch. Twisted pair wire is admirably suited to the task. The higher data rates of coaxial cable or fiber would overwhelm the switches available with today's technology.

1.4 Types of Local Networks

Three categories of local networks can be distinguished: local area network (LAN), high-speed local network (HSLN), and computerized branch exchange (CBX). These categories reflect differences in types of applications, possible choices of topologies and media, and, especially in the case of the CBX, differences in technology. Table 3 is a summary of representative characteristics of each category. Although we have included the CBX as a category of local network because it is an alternative means of locally interconnecting digital devices, its technology and architecture are so different from that of the LAN and HSLN that it is often not considered a local network. We mention its characteristics briefly, but in the remainder of this paper we shall concentrate on the LAN and HSLN.

1.4.1 Local Area Network

The term "local area network" (LAN) refers to a general-purpose local network that can serve a wide variety of devices. LANs support minicomputers, mainframes, terminals, and other peripheral devices. In many cases, these networks can carry not only data, but voice, video, and graphics.

The most common type of LAN is a bus or tree using coaxial cable; rings using twisted pair wire, coaxial cable, or optical

Table 3. Classes of Local Networks

Characteristics	Local area network	High-speed local network	Computerized branch exchange
Transmission medium	Twisted pair, coaxial (both), fiber	CATV coaxial	Twisted pair
Topology	Bus, tree, ring	Bus	Star
Transmission speed (Mbps)	1–20	50	0.0096–0.064
Maximum distance (kilometers)	~25	1	1
Switching technique	Packet	Packet	Circuit
Number of devices supported	100's–1000's	10's	100's–1000's
Attachment cost ($)	500–5000	40,000–50,000	250–1000

fiber are an alternative. The data transfer rates on LANs (1–10 Mbps) are high enough to satisfy most requirements and provide sufficient capacity to allow a large numbers of devices to share the network.

The LAN is probably the best choice when a variety of devices and a mix of traffic types are involved. The LAN, alone or as part of a hybrid local network with one of the other types, will become a common feature of many office buildings and other installations.

1.4.2 High-Speed Local Network

The high-speed local network (HSLN) is designed to provide high end-to-end throughput between expensive high-speed devices, such as mainframes and mass storage devices.

Although other media and topologies are possible, the bus topology using coaxial cable is the most common. Very high data rates are achievable—50 Mbps is standard—but both the distance and the number of devices are more limited in the HSLN than in the LAN.

The HSLN is typically found in a computer room setting. Its main function is to provide I/O channel connections among a number of devices. Typical uses include file and bulk data transfer, automatic backup, and load leveling. Because of the current high prices for HSLN attachment, they are generally not practical for microcomputers and less expensive peripheral devices, and are only rarely used for minicomputers.

1.4.3 Computerized Branch Exchange

The computerized branch exchange (CBX) is a digital on-premise private branch exchange designed to handle both voice and data connections, usually implemented with a star topology using twisted pair wire to connect end points to the switch.

In contrast to the LAN and HSLN, which use packet switching, the CBX uses circuit switching. Data rates to individual end points are typically low, but bandwidth is guaranteed and there is essentially no network delay once a connection has been made. The CBX is well suited to voice traffic, and to both terminal-to-terminal and terminal-to-host data traffic.

2. THE BUS/TREE TOPOLOGY

The bus and tree topologies have been most popular to date in implementing both LANs and HSLNs. Well-known bus or bus/tree architectures include Ethernet [Metcalfe and Boggs 1976], one of the earliest local networks; HYPERchannel [Christensen 1979], the oldest and most popular HSLN; and MITRENET [Hopkins 1979], the basis of much United States government-sponsored research. Most of the low-cost, twisted pair LANs for microcomputers use a bus topology. In this section we describe key characteristics of bus and tree configurations, and present a detailed description of the two transmission techniques used in these configurations: baseband and broadband.

2.1 Characteristics of Bus/Tree LANs and HSLNs

The bus/tree topology is a multipoint configuration; that is, it allows more than two devices at a time to be connected to and capable of transmitting on the medium.

This is opposed to the other topologies that we have discussed, which are point-to-point configurations; that is, each physical transmission link connects only two devices. Because multiple devices share a single data path, only one device may transmit data at a time, usually in the form of a packet containing the address of the destination. The packet propagates throughout the medium and is received by all other stations, but is copied only by the addressed station.

Two transmission techniques are in use for bus/tree LANs and HSLNs: baseband and broadband. Baseband uses digital signaling and can be employed on twisted pair wire or coaxial cable. Broadband uses analog signaling in the radio frequency (RF) range and is employed only on coaxial cable. Some of the differences between baseband and broadband are highlighted in Table 4, and in the following two sections we explore these techniques in some detail. There is also a variant known as "single-channel broadband," which has the signaling characteristics of broadband but some of the restrictions of baseband; this technique is also covered below.

The multipoint nature of the bus/tree topology gives rise to several problems common to both baseband and broadband systems. First, there is the problem of determining which station on the medium may transmit at any point in time. Historically, the most common access scheme has been centralized polling. One station has the role of a central controller. This station may send data to any other station or may request that another station send data to the controller. This method, however, negates some of the advantages of a distributed system and also is awkward for communication between two noncontroller stations. A variety of distributed strategies, referred to as "medium access control protocols," have now been developed for bus and tree topologies. These are discussed in Section 4.

A second problem with the multipoint configuration has to do with signal balancing. When two devices exchange data over a link, the signal strength of the transmitter must be adjusted to be within certain limits.

Table 4. Bus/Tree Transmission Techniques

Baseband	Broadband
Digital signaling	Analog signaling (requires RF modem)
Entire bandwidth consumed by signal—no FDM	FDM possible—multiple data channels, video, audio
Bidirectional	Unidirectional
Bus topology	Bus or tree topology
Distance, up to a few kilometers	Distance: up to tens of kilometers

The signal must be strong enough so that, after attenuation across the medium, it meets the receiver's minimum signal strength requirements, and is strong enough to maintain an adequate signal-to-noise ratio. On the other hand, if it is too strong, the signal will overload the circuitry of the transmitter, which creates harmonics and other spurious signals. Although it is easily done for a point-to-point link, signal balancing for a multipoint configuration is complex. If any device can transmit to any other device, then the signal balancing must be performed for all permutations of stations taken two at a time. For n stations that works out to $nx(n - 1)$ permutations. For example, for a 200-station network (not a particularly large system), 39,800 signal strength constraints must be staisfied simultaneously. With interdevice distances ranging from tens to possibly thousands of meters, this can become an impossible task. In systems that use RF signals, the problem is compounded, owing to the possibility of RF signal interference across frequencies. The solution to these difficulties is to divide the medium into segments within which pairwise balancing is possible (i.e., signals between pairs of devices can be balanced within each segment). Amplifiers or repeaters are used between segments to maintain signal strength.

2.2 Baseband Systems

A baseband LAN or HSLN is by definition one that uses digital signaling. The entire frequency spectrum of the medium is used to form the digital signal, which is inserted on the line as constant-voltage pulses. Baseband systems can extend only about a

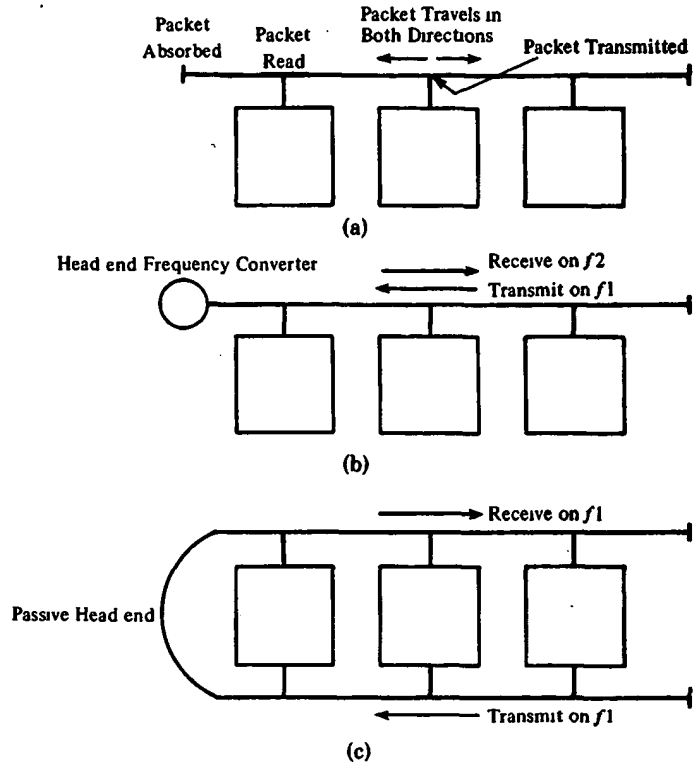


Figure 3. Baseband and broadband transmission: (a) bidirectional (baseband, single-channel broadband); (b) mid-split broadband; (c) dual cable broadband. [From W. Stallings, *Local Networks: An Introduction*, Macmillan, New York, 1984. Copyright Macmillan 1984.]

kilometer at most because the attenuation of the signal, especially at higher frequencies, causes a blurring of the pulses and a weakening of the signal to the extent that communication over longer distances is impractical. Baseband transmission is bidirectional; that is, a signal inserted at any point on the medium propagates in both directions to the ends, where it is absorbed (Figure 3a). In addition, baseband systems require a bus topology. Unlike analog signals, digital signals cannot easily be propagated through the splitters and joiners of a tree topology.

2.2.1 Baseband Coaxial

The most well-known form of baseband bus LAN uses coaxial cable. Unless otherwise indicated, this discussion is based on the Ethernet system [DEC 1983; Metcalfe and Boggs 1976; Metcalfe et al. 1977; Shoch et al. 1983] and the almost-identical IEEE standard [IEEE 1983].

Most baseband coaxial systems use a special 50-ohm cable rather than the more common 75-ohm cable used for cable television and broadband LANs. For digital signals, the 50-ohm cable suffers less intense reflections from the insertion capacitance of the taps, and provides better protection against low-frequency electromagnetic noise.

The simplest baseband coaxial LAN consists of an unbranched length of coaxial cable with a terminator at each end to prevent reflections. A maximum length of 500 meters is recommended. Stations attach to the cable by means of a tap. Attached to each tap is a transceiver, which contains the electronics for transmitting and receiving. The distance between any two taps should be a multiple of 2.5 meters

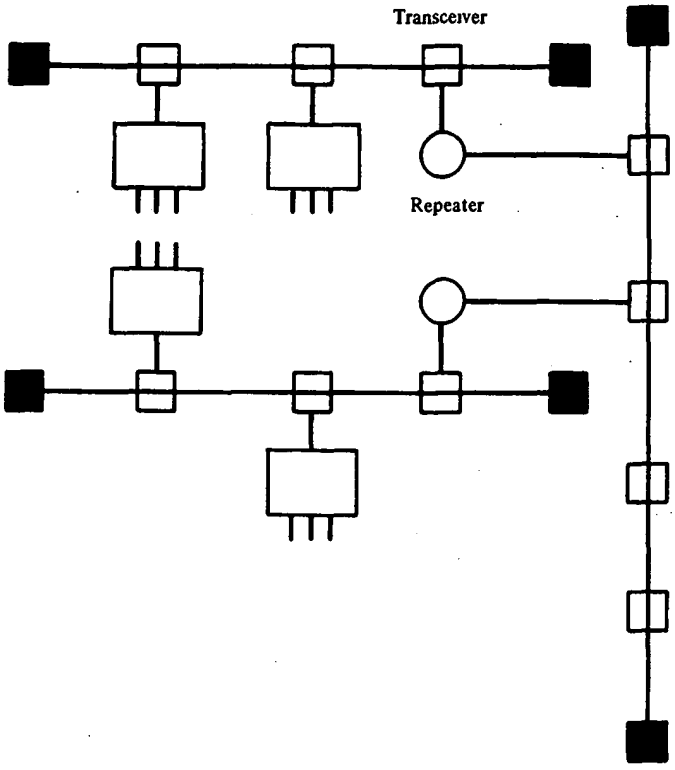


Figure 4. Baseband system.

to ensure that reflections from adjacent taps do not add in phase [Yen and Crawford 1983], and a maximum of 100 taps is recommended.

The above specifications are for a 10-Mbps data rate. They are based on engineering trade-offs involving data rate, cable length, number of taps, and the electrical characteristics of the transmit and receiver components. For example, at lower data rates the cable could be longer.

To extend the length of the network, a repeater may be used. A repeater consists, in essence, of two transceivers joined together and connected to two different segments of coaxial cable. The repeater passes digital signals in both directions between the two segments, amplifying and regenerating the signals as they pass through. A repeater is transparent to the rest of the system; since it does no buffering, it in no sense isolates one segment from another. So, for example, if two stations on different segments attempt to transmit at the same time, their packets will interfere with each other (collide). To avoid multipath interference, only one path of segments and repeaters is allowed between any two stations. Figure 4 is an example of a baseband system with three segments and two repeaters.

2.2.2 *Twisted Pair Baseband*

A twisted pair baseband LAN is intended for low-cost, low-performance requirements. Although this type of system supports fewer stations at lower speeds than a coaxial baseband LAN, its cost is also far lower.

The components of the system are few and simple:

- twisted pair bus,
- terminators,
- controller interface.

The latter can simply be a standard two-wire I/O or communications interface.

Typically, the electrical signaling technique on the cable conforms to RS-422-A. This is a standard inexpensive interface.

With this kind of network, the following parameters are reasonable:

- length, up to 1 kilometer,
- data rate, up to 1 Mbps,
- number of devices, 10's.

For these requirements, twisted pair wire is a good medium for several reasons: It is lower in cost than coaxial cable and provides equal noise immunity, and virtually anyone can install the network. The task of installation consists of laying the cable and connecting the controllers. This requires only a screwdriver and a pair of pliers, and is similar to hooking up hi-fi speakers.

Examples of these systems can be found in Malone [1981], Bosen [1981], and Hahn and Belanger [1981].

2.3 Broadband Systems

Within the context of local networks, a broadband system is by definition one that uses analog signaling. Unlike digital signaling, in which the entire frequency spectrum of the medium is used to produce the signal, analog signaling allows frequency-division multiplexing (FDM). With FDM, the frequency spectrum on the cable is divided into channels or sections of bandwidth; separate channels can support data traffic, TV, and radio signals.

A special case of a broadband system is a low-cost system using only a single channel. To distinguish between the two cases, we will refer to "FDM broadband" and "single-channel broadband" in this section. In Section 2.3.1, we examine FDM broadband, describing two different physical configurations, dual cable and split cable, and three different data transfer techniques, dedicated, switched, and multiple access. In Section 2.3.2, we describe single-channel broadband.

2.3.1 FDM Broadband

Broadband systems use standard, off-the-shelf community antenna television (CATV) components, including 75-ohm coaxial cable. All end points are terminated with a 75-ohm terminator to absorb signals. Broadband components allow splitting and joining operations; hence both bus and tree topologies are possible. Broadband is suitable for a range of tens of kilometers and hundreds or even thousands of devices. For all but very short distances, amplifiers are required.

A typical broadband system is shown in Figure 5. As with baseband, stations attach to the cable by means of a tap. Unlike baseband, however, broadband is a unidirectional medium; signals inserted onto the medium can only propagate in one direction. The primary reason for this is that it is not feasible to build amplifiers that will pass signals of one frequency in both directions. This unidirectional property means that only those stations "downstream" from a transmitting station can receive its signals. How, then, can one achieve full connectivity?

Clearly, two data paths are needed. These paths are joined at a point on the network known as the head end. For the bus topology, the head end is simply one end of the bus; for the tree topology, the head end is the root of the branching tree. All stations transmit on one path toward the head end (inbound). Signals received at the head end then are propagated along a second data path away from the head end (outbound). All stations receive on the outbound path.

Physically, two different configurations are used to implement the inbound and outbound paths (see Figure 3). On a dual cable configuration, the inbound and outbound paths are separate cables, the head end being simply a passive connector between the two. Stations send and receive on the same frequency.

In contrast, in the split configuration, the inbound and outbound paths are different frequencies on the same cable. Bidirectional amplifiers pass lower frequencies inbound and higher frequencies outbound. The head end contains a device, known as a frequency converter, for translating inbound frequencies to outbound frequencies.

The frequency converter at the head end can be either an analog or a digital device. An analog device simply translates signals

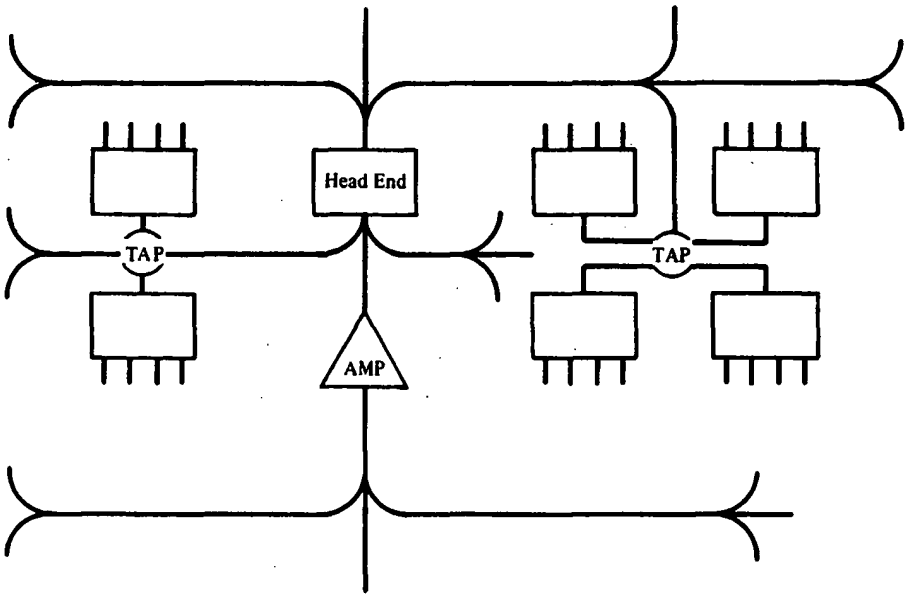


Figure 5. Broadband system.

to a new frequency and retransmits them. A digital device recovers the digital data from the head end and then retransmits the cleaned-up data on the new frequency.

Split systems are categorized by the frequency allocation to the two paths. Subsplit, commonly used by the CATV industry, provides 5–30 megahertz (MHz) inbound and 40–300 MHz outbound. This system was designed for metropolitan area TV distribution, with limited subscriber-to-central office communication. Mid-split, more suitable for LANs, provides an inbound range of 5–116 MHz and an outbound range of 168–300 MHz. This provides a more equitable distribution of bandwidth. Midsplit was developed at a time when the practical spectrum of a CATV cable was 300 MHz. Spectrums surpassing 400 MHz are now available, and either "supersplit" or "equal-split" is sometimes used to achieve even better balance by splitting the bandwidth roughly in half.

The differences between split and dual are minor. The split system is useful when a single-cable plant is already installed in a building, and the installed system is about 10–15 percent cheaper than a dual cable system [Hopkins 1979]. On the other hand, a dual cable has over twice the capacity of mid-split, and does not require the frequency translator at the head end. A further comparison can be found in Cooper and Edholm [1983].

The broadband LAN can be used to carry multiple channels, some used for analog signals, such as video and voice, and some for data. For example, a video channel requires a 6-MHz bandwidth. Digital channels can generally carry a data rate of somewhere between 0.25 and 1 bit per second per hertz. A possible allocation of a 350-MHz cable is shown in Figure 6.

One of the advantages of the use of multiple channels is that different channels can be used to satisfy different requirements. Figure 7 depicts three kinds of data transfer techniques that can be used: dedicated, switched, and multiple access.

For dedicated service, a small portion of the cable's bandwidth is reserved for exclusive use by two devices. No special protocol is needed. Each of the two devices attaches to the cable through a modem; both modems are tuned to the same frequency. This technique is analogous to securing a dedicated leased line from the telephone company. Transfer rates of up to 20 Mbps are achievable. The dedicated service could be used to connect two devices when a heavy

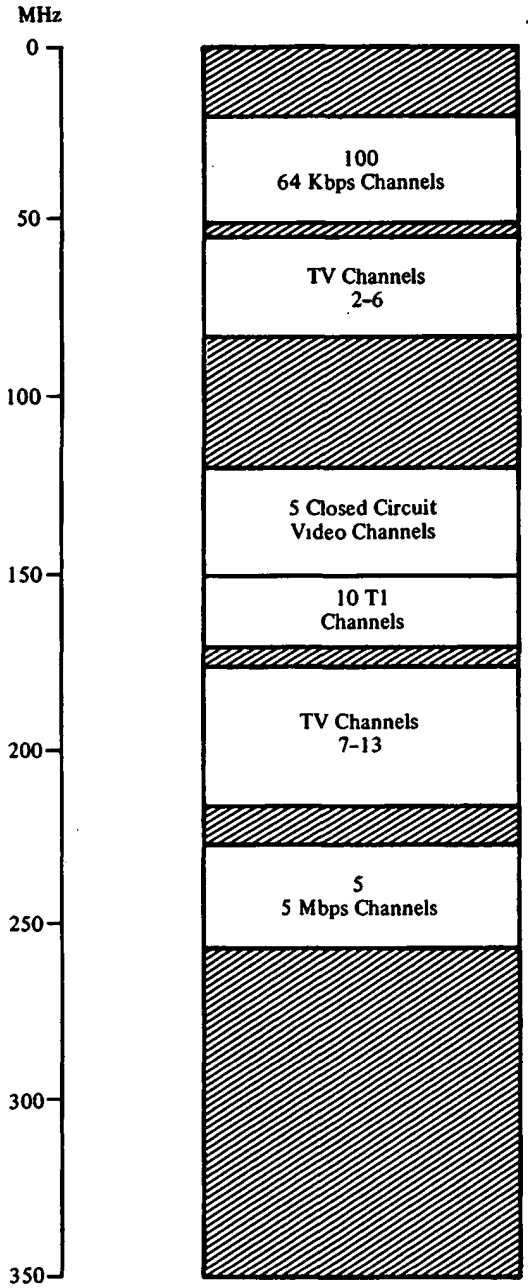


Figure 6. Broadband spectrum allocation.

traffic pattern is expected; for example, one computer may act as a standby for another, and need to get frequent updates of state information and file and database changes.

The switched transfer technique requires the use of a number of frequency bands. Devices are attached through "frequency-agile" modems, capable of changing their frequency by electronic command. All attached devices, together with a controller, are initially tuned to the same frequency. A station wishing to establish a connection sends a request to the controller, which assigns an available frequency to the two devices and signals their modems to tune to that frequency. This technique is analogous to a dial-up line. Because the cost of frequency-agile modems rises dramatically with data rate, rates of 56 Kbps or less are typical. The switched technique is used in Wang's local network for terminal-to-host connections [Stahlman 1982], and could also be used for voice service.

Finally, the multiple-access service, by far the most common, allows a number of attached devices to be supported at the same frequency. As with baseband, some form of medium access control protocol is needed to control transmission. It provides for distributed, peer communications among many devices, which is the primary motivation for a local network.

Further discussions of FDM broadband LANs can be found in Cooper [1982, 1983], Dineson and Picazo [1980], and Forbes [1981].

2.3.2 *Single-Channel Broadband*

An abridged form of broadband is possible, in which the entire spectrum of the cable is devoted to a single transmission path for analog signals. In general, a single-channel broadband LAN uses bidirectional transmission and a bus topology. Hence there can be no amplifiers, and there is no need for a head end. Transmission is generally at a low frequency (a few megahertz). This is an advantage since attenuation is less at lower frequencies.

Because the cable is dedicated to a single task, it is not necessary to ensure that the modem output is confined to a narrow bandwidth; energy can spread over the cable's spectrum. As a result, the electronics are simple and inexpensive. This scheme would appear to give comparable performance, at a comparable price, to baseband.

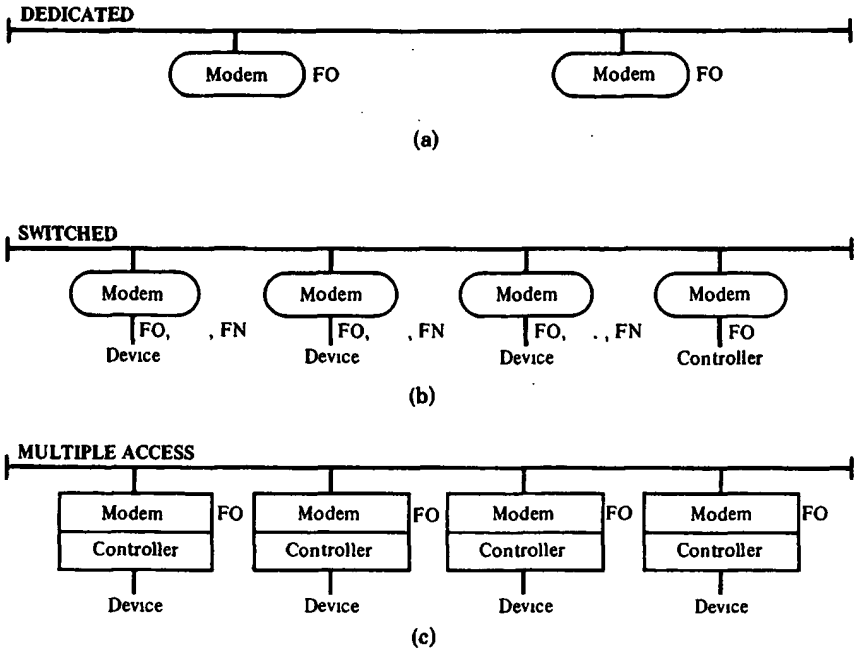


Figure 7. Broadband data transfer services: (a) dedicated; (b) switched; (c) multiple access.

The principle reason for choosing this scheme over baseband appears to be that single-channel broadband uses 75-ohm cable. Thus a user with modest initial requirements could install the 75-ohm cable and use inexpensive single-channel components. Later, if the needs expand, the user could switch to full broadband by replacing modems, but would avoid the expense of rewiring the building with new cable.

2.4 Baseband versus Broadband

One of the least productive aspects of the extensive coverage afforded local networks in the trade and professional literature is the baseband versus broadband debate. The potential customer will be faced with many other decisions more complex than this one. The fact is that there is room for both technologies in the local network field. For the interested reader, thoughtful discussions may be found in Hopkins and Meisner [1982] and Krutsch [1981].

To briefly summarize the two technologies, baseband has the advantages of simplicity, and, in principle, low cost. The layout of a baseband cable plant is simple: A relatively inexperienced local network engineer should be able to cope with it. Baseband's potential disadvantages include limitations in capacity and distance—but these are only disadvantages if one's requirements exceed those limitations.

Broadband's strength is its tremendous capacity. Broadband can carry a wide variety of traffic on a number of channels, and with the use of active amplifiers can achieve very wide area coverage. Also, the system is based on a mature CATV technology, with reliable and readily available components. A disadvantage of broadband systems is that they are more complex than baseband to install and maintain, requiring experienced RF engineers. Also, the average propagation delay between stations for broadband is twice that for a comparable baseband system. This reduces the efficiency and performance of the system, as discussed in Section 5.

As with all other network design choices, the selection of baseband or broadband

must be based on the relative cost and benefits. Neither is likely to win the LAN war.

2.5 Bus HSLNs

HSLNs using a bus topology share a number of characteristics with bus-based LANs. In particular, data are transmitted in packets, and because the configuration is multipoint, a medium access control protocol is needed. From the user's point of view, the key difference between a LAN and an HSLN is the higher data rate of the latter. Commercial HSLN products and the proposed American National Standards Institute (ANSI) standard both specify 50 Mbps, whereas 10 Mbps or less is typical for LANs. The higher data rate of the HSLN imposes cost and technical constraints, and the HSLN is limited to just a few devices (10–20) over a relatively small distance (less than 1 kilometer).

Two techniques have been used for bus HSLNs: baseband and single-channel broadband. In both cases a 75-ohm coaxial cable is used. Baseband is used for the most widely available HSLN product, HYPERchannel [Christensen 1979; Thornton 1980]. Single-channel broadband is used on CDC's product [Hohn 1980] and is the approach taken in the proposed ANSI standard [Burr 1983].

It is difficult to assess the relative advantages of these two schemes. In general, the two approaches should give comparable performance at comparable cost.

3. THE RING TOPOLOGY

The major alternative to the bus/tree topology LAN is the ring. The ring topology has been popular in Europe but is only slowly gaining ground in the United States, where Ethernet and MITRENET were largely responsible for shaping the early direction of activity. Several factors suggest that the ring may become more of a competitor in the United States: IBM has conducted research on ring LANs and is expected to announce a product, and the IEEE 802 ring LAN standard is moving toward completion.

3.1 Characteristics of Ring LANs

A ring LAN consists of a number of repeaters, each connected to two others by unidirectional transmission links to form a single closed path. Data are transferred sequentially, bit by bit, around the ring from one repeater to the next. Each repeater regenerates and retransmits each bit.

For a ring to operate as a communications network, three functions are required: data insertion, data reception, and data removal. These functions are provided by the repeaters. Each repeater, in addition to serving as an active element on the ring, serves as an attachment point for a station. Data are transmitted in packets, each of which contains a destination address field. As a packet circulates past a repeater, the address field is copied. If the attached station recognizes the address, then the remainder of the packet is copied. A variety of strategies can be used for determining how and when packets are added to and removed from the ring. These strategies are medium access control protocols, which are discussed in Section 4.

Repeaters perform the data insertion and reception functions in a similar manner to taps, which serve as station attachment points on a bus or tree. Data removal, however, is more difficult on a ring. For a bus or tree, signals inserted onto the line propagate to the end points and are absorbed by terminators, which clear the bus or tree of data. However, because the ring is a closed loop, data will circulate indefinitely unless removed. There are two solutions to this problem: A packet may be removed by the addressed repeater when it is received, or each packet could be removed by the transmitting repeater after it has made one trip around the ring. The latter approach is more desirable because: (1) it permits automatic acknowledgment, and (2) it permits multicast addressing, that is, one packet sent simultaneously to multiple stations.

The repeater then can be seen to have two main purposes: (1) to contribute to the proper functioning of the ring by passing on all the data that comes its way, and (2) to provide an access point for attached stations to send and receive data. Correspond-

ing to these two purposes are two states (Figure 8): listen and transmit.

In the listen state, each bit that is received by the repeater is retransmitted, with a small delay to allow the repeater to perform required functions. Ideally, this delay should be on the order of one bit time (the time it takes for a repeater to transmit one complete bit onto the outgoing line). The required functions are

(1) Scan the passing bit stream for pertinent patterns, chiefly the address or addresses of attached stations. Another pattern, used in the token control strategy, is discussed in Section 4. Note that to perform the scanning function, the repeater must have some knowledge of packet format.
(2) Copy each incoming bit and send it to the attached station, while continuing to retransmit each bit. This will be done for each bit of each packet that is addressed to this station.
(3) Modify a bit as it passes by. In certain control strategies, bits may be modified, for example, to indicate that the packet has been copied. This would serve as an acknowledgment.

When a repeater's station has data to send and the repeater has permission to transmit (based on the medium access control strategy), the repeater enters the transmit state. In this state, the repeater receives bits from the station and retransmits them on its outgoing link. During the period of transmission, bits may appear on the incoming ring link. There are two possibilities, and they are treated differently:

(1) The bits could be from the same packet that the repeater is still sending. This will occur if the "bit length" of the ring is shorter than the packet. In this case, the repeater passes the bits back to the station, which can check them as a form of acknowledgment.
(2) For some control strategies, more than one packet could be on the ring at the same time. If the repeater, while transmitting, receives bits from a packet that it did not originate, it must buffer them to be transmitted later.

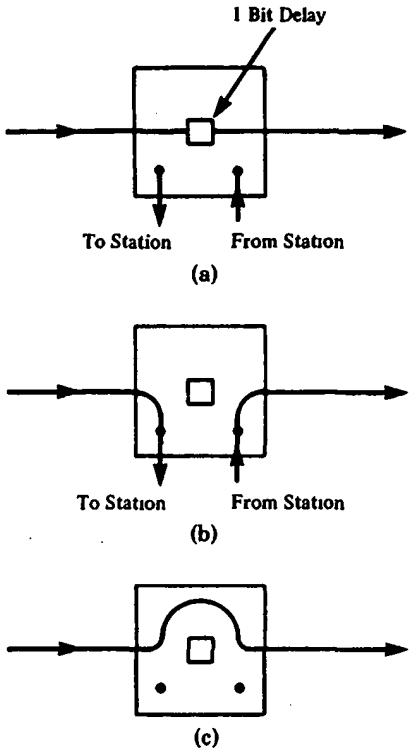


Figure 8. Ring repeater states: (a) listen state; (b) transmit state; (c) bypass state.

These two states, listen and transmit, are sufficient for proper ring operation. A third state, the bypass state, is also useful. In this state, a bypass relay can be activated, so that signals propagate past the repeater with no delay other than medium propagation. The bypass relay affords two benefits: (1) it provides a partial solution to the reliability problem, discussed later, and (2) it improves performance by eliminating repeater delay for those stations that are not active on the network.

Twisted pair wire, baseband coaxial cable, and fiber optic cable all can be used to provide the repeater-to-repeater links. Broadband coaxial cable could not be used easily, as each repeater would have to be capable of receiving and transmitting data on multiple channels asynchronously.

3.2 Potential Ring Problems

One of the principal reasons for the slow acceptance of the ring LAN in the United

States is that there are a number of potential problems with this topology [Saltzer and Pogran 1979]:

Cable Vulnerability. A break on any of the links between repeaters disables the entire network until the problem can be isolated and a new cable installed.

Repeater Failure. A failure of a single repeater also disables the entire network. In many networks, some of the stations may not be in operation at any given time; yet all repeaters must always operate properly.

Perambulation. When either a repeater or a link fails, locating the failure requires perambulation of the ring, and thus access to all rooms containing repeaters and cable. This is known as the "pocket full of keys" problem.

Installation Headaches. Installation of a new repeater to support new devices requires the identification of two topologically adjacent repeaters near the new installation. It must be verified that they are in fact adjacent (documentation could be faulty or out of date), and cable must be run from the new repeater to each of the old repeaters. There are several possible consequences: The length of cable driven by the source repeater may change, possibly requiring returning; old cable will accumulate if not removed; and the geometry of the ring may become highly irregular, exacerbating the perambulation problem.

Size Limitations. There is a practical limit to the number of stations on a ring. This limit is suggested by the reliability and maintenance problems cited earlier and by the accumulating delay of large numbers of repeaters. A limit of a few hundred stations seems reasonable.

Initialization and Recovery. To avoid designating one ring node as a controller (negating the benefit of distributed control), a strategy is required to ensure that all stations can cooperate smoothly when initialization and recovery are required. This cooperation is needed, for example, when a packet is garbled by a transient line error; in that case, no repeater may wish to assume the responsibility of removing the circulating packet.

The last problem is a protocol issue, which we discuss later. The remaining problems can be handled by a refinement of the ring topology, discussed next.

3.3 An Enhanced Architecture

The potential ring problems cited above have led to the development of an enhanced ring architecture that overcomes these problems of the ring and allows the construction of large local networks. This architecture is the basis of IBM's anticipated local network product [Rauch-Hindin 1982] and grows out of research done at IBM [Bux et al. 1982] and Massachusetts Institute of Technology [Saltzer and Clark 1981]. The result is a ring-based local network with two additions: ring wiring concentrators and bridges.

The ring wiring concentrator is simply a centralized location through which interrepeater links are threaded. The link between any two stations runs from one station into the concentrator and back out to the second station. This technique has a number of advantages. Because there is central access to the signal on every link, it is a simple matter to isolate a fault. A message can be launched into the ring and tracked to see how far it gets without mishap. A faulty segment can be disconnected and repaired at a later time. New repeaters can easily be added to the ring: Simply run two cables from the new repeater to the site of ring wiring concentration and splice into the ring.

The bypass relay associated with each repeater can be moved into the ring wiring concentrator. The relay can automatically bypass its repeater and two links for any malfunction. A nice effect of this feature is that the transmission path from one working repeater to the next is approximately constant; thus the range of signal levels to which the transmission system must automatically adapt is much smaller.

The ring wiring concentrator permits rapid recovery from a cable or repeater failure. Nevertheless, a single failure could temporarily disable the entire network. Furthermore, throughput considerations

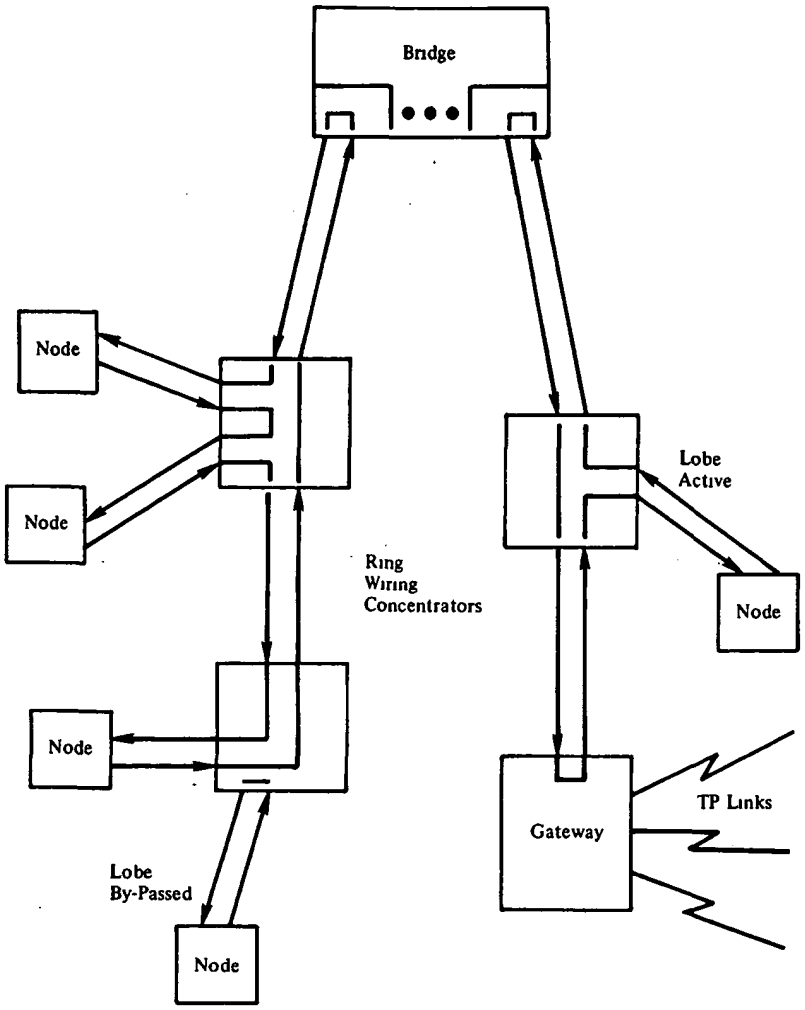


Figure 9. Star-ring architecture.

place a practical upper limit on the number of stations in a ring, since each repeater adds an increment of delay. Finally, in a spread-out network, a single wire concentration site dictates a lot of cable.

To attack these remaining problems, consider a local network consisting of multiple rings (Figure 9). Each ring consists of a connected sequence of wiring concentrators; the set of rings is connected by a bridge. The bridge routes data packets from one ring subnetwork to another on the basis of addressing information in the packet so routed. From a physical point of view, each ring operates independently of the other rings attached to the bridge. From a logical point of view, the bridge provides transparent routing between the two rings.

The bridge must perform five functions:

Input Filtering. For each ring, the bridge monitors the traffic on the ring and copies all packets addressed to other rings on the bridge. This function can be performed by a repeater programmed to recognize a family of addresses rather than a single address.

Input Buffering. Received packets may need to be buffered, either because the interring traffic is peaking, or because the output buffer at the destination ring is temporarily full.

Switching. Each packet must be routed through the bridge to its appropriate destination ring.

Output Buffering. A packet may need to be buffered at the threshold of the destination ring, waiting for an opportunity to be inserted.

Output Transmission. This function can be performed by an ordinary repeater.

Two principal advantages accrue from the use of a bridge. First, the failure of a ring, for whatever reason, will disable only a portion of the network; failure of the bridge does not prevent intraring traffic. Second, multiple rings may be employed to obtain a satisfactory level of performance when the throughput capability of a single ring is exceeded.

There are several disadvantages to be noted. First, the automatic acknowledgment feature of the ring is lost; higher level protocols must provide acknowledgment. Second, performance may not significantly improve if there is a high percentage of interring traffic. If it is possible to do so, network devices should be judiciously allocated to rings to minimize interring traffic.

3.4 Bus versus Ring

For the user with a large number of devices and high capacity requirements, the bus or tree broadband LAN seems the best suited to the requirements. For more moderate requirements, the choice between a baseband bus LAN and a ring LAN is not at all clear-cut.

The baseband bus is the simpler system. Passive taps rather than active repeaters are used. Thus medium failure is less likely, and there is no need for the complexity of bridges and ring wiring concentrators.

The most important benefit of the ring is that, unlike the bus/tree, it uses point-to-point communication links, which has a number of implications. First, since the transmitted signal is regenerated at each node, transmission errors are minimized and greater distances can be covered. Second, the ring can accommodate optical fiber links, which provide very high data rates and excellent electromagnetic interference (EMI) characteristics. Third, the electronics and maintenance of point-to-point lines are simpler than those for multipoint lines. Another benefit of the ring is, assuming that the enhanced ring architecture is used, that fault isolation and recovery is simpler than for bus/tree.

Further discussion of ring versus bus is contained in Salwen [1983].

4. MEDIUM ACCESS CONTROL PROTOCOLS

All local networks (LAN, HSLN, CBX) consist of a collection of devices that must share the network's transmission capacity. Some means of controlling access to the transmission medium is needed so that any two particular devices can exchange data when required.

The key parameters in controlling access to the medium are "where" and "how." The question of where access is controlled generally refers to whether the system is centralized or distributed. A centralized scheme has the advantages of

- possibly affording greater control over access by providing such things as priorities, overrides, and guaranteed bandwidth,
- allowing the logic at each station to be as simple as possible, and
- avoiding problems of coordination,

and the disadvantages of

- a single point of failure, and
- possibly acting as a bottleneck, reducing efficiency.

The pros and cons for distributed control are mirror images of the above points.

How access is controlled is constrained by the topology and is a trade-off among competing factors: cost, performance, and complexity.

The most common medium access control protocols for local networks are categorized in Figure 10. In all cases, multiple data transfers share a single transmission medium. This always implies some sort of multiplexing, either in the time or frequency domain. In the frequency domain, any technique on a multiple-channel broadband system is by definition based on frequency-division multiplexing (FDM).

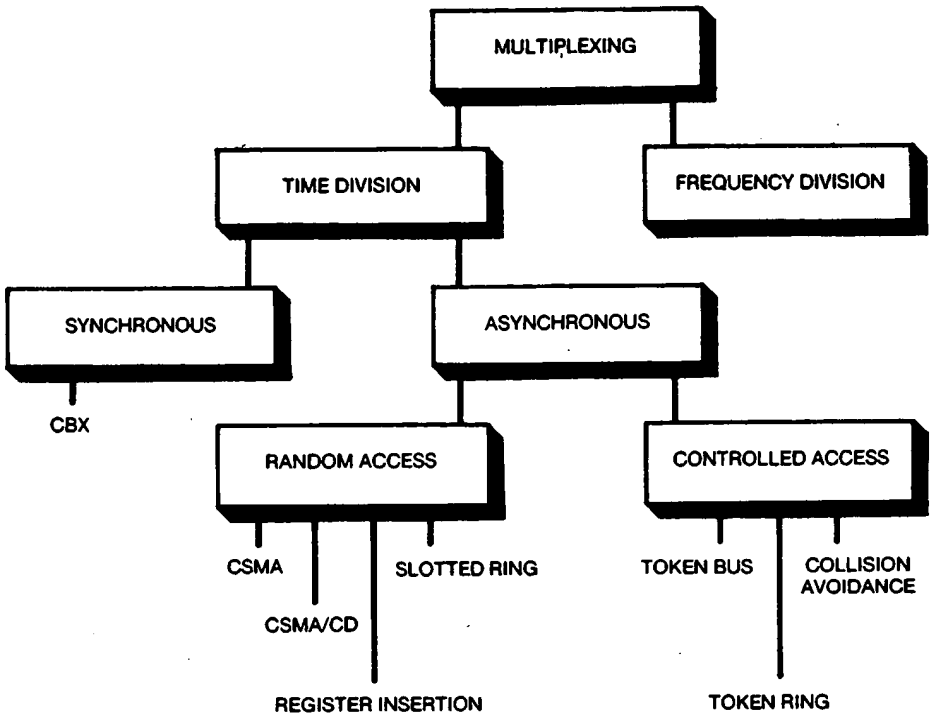


Figure 10. Local network access control techniques.

Within a single channel, however, some form of time-division multiplexing is required.

Time-division access control techniques are either synchronous or asynchronous. With synchronous techniques, a specific capacity is dedicated to a connection, as, for example, in the CBX. This is not optimal in bus/tree and ring networks: The needs of the stations are unpredictable, and the transmission capacity should be allocated in an asynchronous (dynamic) fashion in response to these needs.

Access to the medium using asynchronous time-division multiplexing (TDM) may be random (stations attempt to access the medium at will and at random times) or regulated (an algorithm is used to regulate the sequence and time of station access). The random access category includes two common bus techniques, carrier sense multiple access (CSMA) and carrier sense multiple access with collision detection (CSMA/CD), and two common ring techniques, register insertion and slotted ring. The regulated access category includes both token bus and token ring for LANs and collision avoidance, the most common HSLN technique. All of the asynchronous techniques have found widespread application in LANs or HSLNs, and are discussed in this section.

4.1 Bus/Tree LANs

Of all the local network topologies, the bus/tree topologies present the most challenges, and the most options, for medium access control. In this section we do not survey the many techniques that have been proposed; good discussions can be found in Luczak [1978] and Franta and Chlamtec [1981]. Rather, emphasis will be on the two techniques for which standards have been developed by the IEEE 802 Committee and that seem likely to dominate the marketplace: CSMA/CD and token bus. Table 5 is a comparison of these two techniques on a number of characteristics.

Table 5. Bus/Tree Access Methods

Characteristic	CSMA/CD	Token bus
Access determination	Contention	Token
Packet length restriction	Greater than 2× propagation delay	None
Principal advantage	Simplicity	Regulated/fair access
Principal disadvantage	Performance under heavy load	Complexity

4.1.1 CSMA/CD

The most commonly used medium access control protocol for bus/tree topologies is CSMA/CD. The original baseband version is seen in Ethernet, and the original broadband version is part of MITRENET [Hopkins 1979; Roman 1977].

We begin the discussion by looking at a simpler version of this technique known as CSMA or listen before talk (LBT). A station wishing to transmit listens to the medium to determine whether another transmission is in progress. If the medium is idle, the station may transmit. Otherwise, the station backs off for some period of time and tries again, using one of the algorithms explained below. After transmitting, a station waits for a reasonable amount of time for an acknowledgment, taking into account the maximum round-trip propagation delay and the fact that the acknowledging station must also contend for the channel in order to respond.

This strategy is effective for systems in which the packet transmission time is much longer than the propagation time. Collisions only occur when more than one user begins transmitting within the period of propagation delay. If there are no collisions during the time that it takes for the leading edge of the packet to propagate to the farthest station, then the transmitting station has seized the channel and the packet is transmitted without collision.

With CSMA, an algorithm is needed to specify what a station should do if the medium is found to be busy. Three approaches or "persistence algorithms," are possible:

(1) *Nonpersistent.* The station backs off a random amount of time and then senses the medium again.

(2) *1-Persistent.* The station continues to sense the medium until it is idle, then transmits.

(3) *p-Persistent.* The station continues to sense the medium until it is idle, then transmits with some preassigned probability p. Otherwise, it backs off a fixed amount of time, then transmits with probability p or continues to back off with probability $(1 - p)$.

The nonpersistent algorithm is effective in avoiding collisions; two stations wishing to transmit when the medium is busy are likely to back off for different amounts of time. The drawback is that there is likely to be wasted idle time following each transmission. In contrast, the 1-persistent algorithm attempts to reduce idle time by allowing a single waiting station to transmit immediately after another transmission. Unfortunately, if more than one station is waiting, a collision is guaranteed! The p-persistent algorithm is a compromise that attempts to minimize both collisions and idle time.

CSMA/CD, also referred to as listen while talk (LWT), attempts to overcome one glaring inefficiency of CSMA: When two packets collide, the medium remains unusable for the duration of transmission of both damaged packets. For packets that are long in comparison to their propagation time, the amount of wasted bandwidth can be considerable. This waste is reduced with the CSMA/CD protocol: A station continues to listen to the medium while it is transmitting. Hence the following rules are added to the CSMA protocol:

(1) If a collision is detected during transmission, immediately cease transmitting the packet and transmit a brief jamming signal to ensure that all stations know there has been a collision.

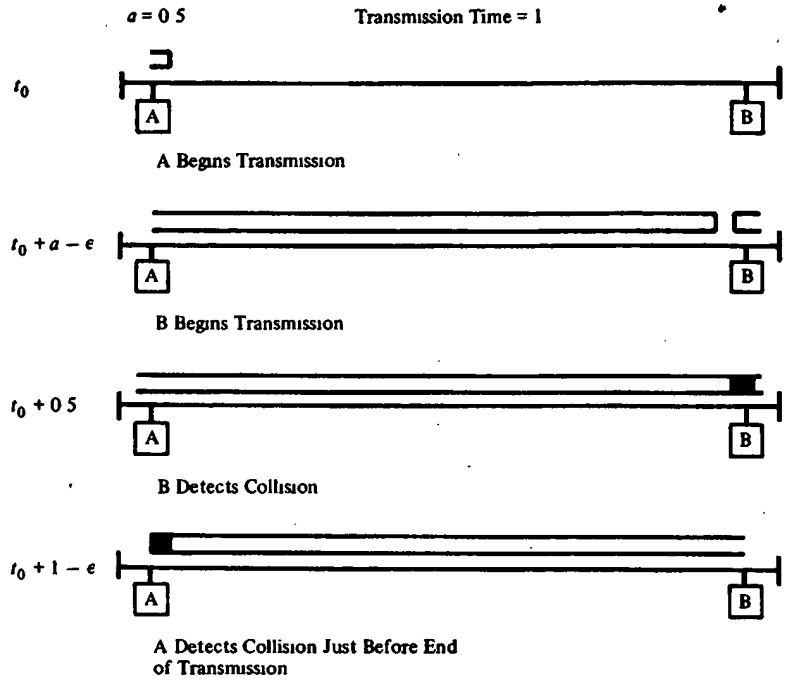


Figure 11. Baseband collision detection timing (transmission time is normalized to 1; a = propagation time). [From W. Stallings, *Local Networks: An Introduction*, Macmillan, New York, 1984. Copyright Macmillan 1984.]

(2) After transmitting the jamming signal, wait a random amount of time, then attempt to transmit again, using CSMA.

When these rules are followed, the amount of wasted bandwidth is reduced to the time that it takes to detect a collision. For proper operation of the algorithm, the minimum packet size should be sufficient to permit collision detection in the worst case. For a baseband system, with two stations that are as far apart as possible (worst case), the time that it takes to detect a collision is twice the propagation delay (Figure 11). For a broadband system, in which the worst case is two stations close together and as far as possible from the head end, the wait is four times the propagation delay from the station to the head end (Figure 12).

As with CSMA, CSMA/CD employs one of the three persistence algorithms. Unexpectedly, the most common choice is 1-persistent; it is used by both Ethernet and MITRENET, and used in the IEEE 802 standard. As was mentioned, the problem with the nonpersistent scheme is the wasted idle time. Although it is more efficient, p-persistent still may result in considerable waste. With the 1-persistent scheme, that waste is eliminated at the cost of wasted collision time.

What saves the day is that the time wasted due to collisions is mercifully short if the packets are long relative to propagation delay. With random back off, two stations involved in a collision are unlikely to collide on their next tries. To ensure stability of this back off, a technique known as binary exponential back off is used. A station attempts to transmit repeatedly in the face of repeated collisions, but the mean value of the random delay is doubled after each collision. After a number of unsuccessful attempts, the station gives up and reports an error.

Although the implementation of CSMA/CD is substantially the same for baseband and broadband, there are a few differences. One difference is the means for performing carrier sense. For baseband systems, the carrier sense operation consists of detecting

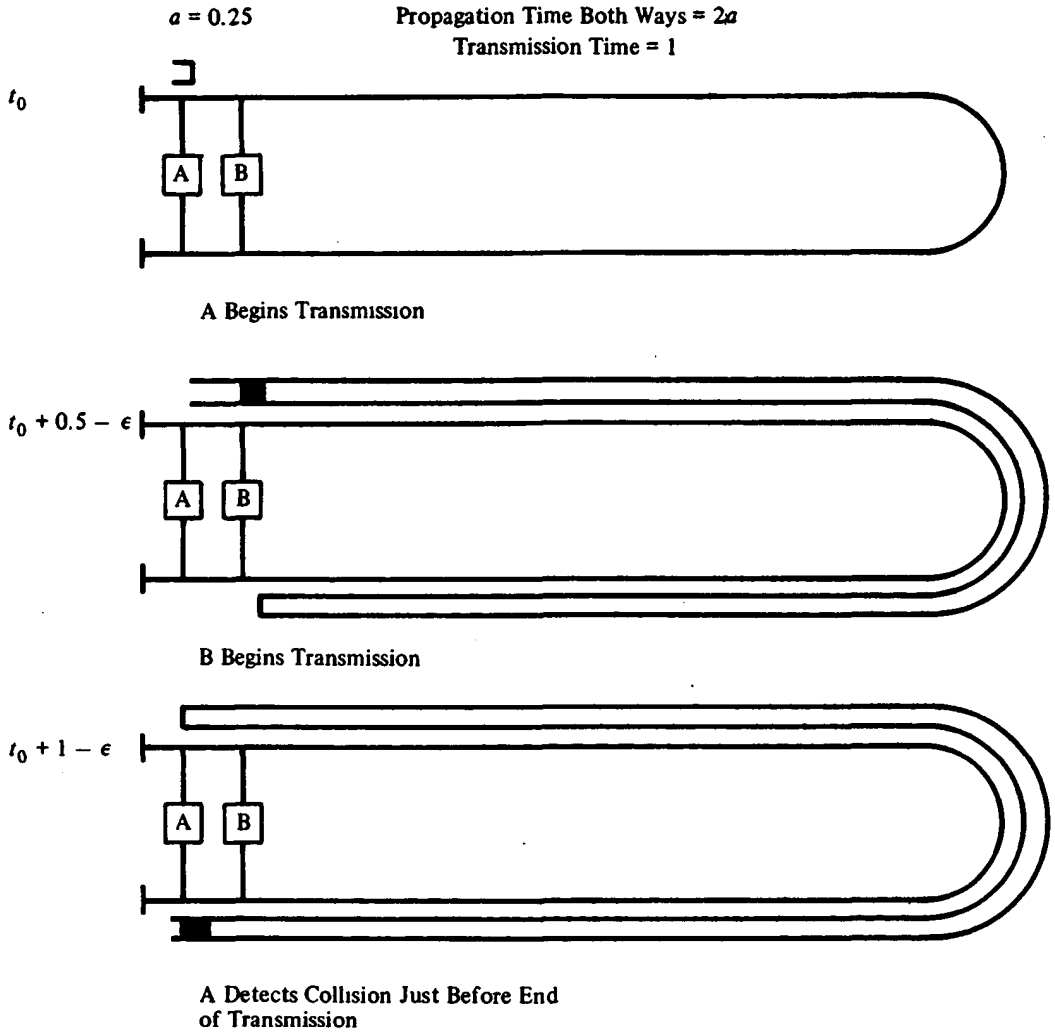


Figure 12. Broadband collision detection timing (transmission time is normalized to 1; a = propagation time one way). [From W. Stallings, *Local Networks: An Introduction*, Macmillan, New York, 1984. Copyright Macmillan 1984.]

a stream of voltage pulses; for broadband, the RF carrier is detected.

Another difference is in collision detection (CD). In a baseband system, a collision produces higher voltage swings than those produced by a single transmitter. A transmitting transceiver detects a collision if the signal on the cable exceeds the maximum that could be produced by the transceiver alone. Attenuation presents a potential problem: If two stations that are far apart are transmitting, each station will receive a greatly attenuated signal from the other. The signal strength could be so small that when added to the transmitted signal at the transceiver, the combined signal does not exceed the CD threshold. This is one reason that Ethernet retricts the maximum length of cable to 500 meters. Because frames may cross repeater boundaries, collisions must cross as well. Hence, if a repeater detects a collision on either cable, it must transmit a jamming signal on the other side.

There are several approaches to collision detection in broadband systems. The most common of these is to perform a bit-by-bit comparison between transmitted and received data. When a station transmits on

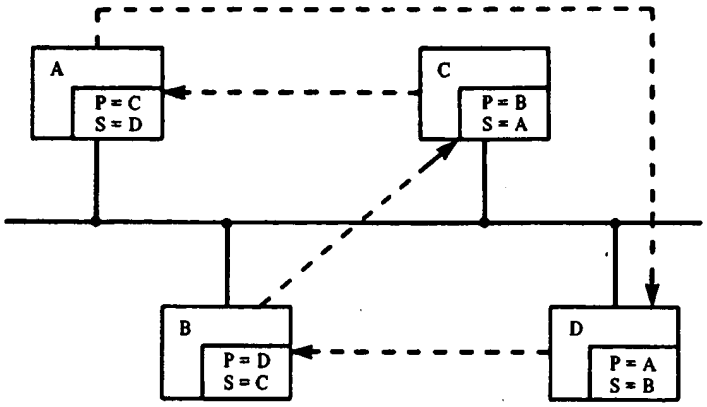


Figure 13. Token bus.

the inbound channel, it begins to hear its own transmission on the outbound channel after a propagation delay to the head end and back. In the MITRE system, the first 16 bits of the transmitted and received signals are compared and a collision is assumed if they differ. There are several problems with this approach. The most serious is the danger that differences in signal level between colliding signals will cause the receiver to treat the weaker signal as noise, and fail to detect the collision. The cable system, with its taps, splitters, and amplifiers, must be carefully tuned so that attenuation effects and differences in transmitter signal strength do not cause this problem. Another problem for dual cable systems is that a station must simultaneously transmit and receive on the same frequency. Its two RF modems must be carefully shielded to prevent cross talk.

An alternative approach for broadband is to perform the CD function at the head end. This is most appropriate for the midsplit system, which has an active component at the head end anyway. This reduces the problem of tuning, ensuring that all stations produce approximately the same signal level at the head end, which then can detect collisions by looking for garbled data or higher-than-expected signal strength.

4.1.2 Token Bus

Token bus is a technique in which the stations on the bus or tree form a logical ring; that is, the stations are assigned positions in an ordered sequence, with the last member of the sequence followed by the first. Each station knows the identity of the stations preceding and following it (Figure 13).

A control packet known as the token regulates the right of access: When a station receives the token, it is granted control of the medium for a specified time, during which it may transmit one or more packets and may poll stations and receive responses. When the station is done, or time has expired, it passes the token on to the next station in logical sequence. Hence steady-state operation consists of alternating data transfer and token transfer phases.

Non-token-using stations are allowed on the bus, but these stations can only respond to polls or requests for acknowledgment. It should also be pointed out that the physical ordering of the stations on the bus is irrelevant and independent of the logical ordering.

This scheme requires considerable maintenance. The following functions, at a minimum, must be performed by one or more stations on the bus:

Addition to Ring. Nonparticipating stations must periodically be granted the opportunity to insert themselves in the ring.

Deletion from Ring. A station can voluntarily remove itself from the ring by splicing together its predecessor and successor.

Fault Management. A number of errors can occur. These include duplicate address (two stations think that it is their turn) and broken ring (no station thinks that it is its turn).

Ring Initialization. When the network is started up, or after the logical ring has broken down, it must be reinitialized. Some cooperative, decentralized algorithm is needed to sort out who goes first, who goes second, etc.

In the remainder of this section we briefly describe the approach taken for these functions in the IEEE 802 standard.

To accomplish *addition to ring*, each node in the ring is responsible for periodically granting an opportunity for new nodes to enter the ring. While holding the token, the node issues a *solicit-successor* packet, inviting nodes with an address between itself and the next node in logical sequence to request entrance. The transmitting node then waits for a period of time equal to one response window or slot time (twice the end-to-end propagation delay of the medium). If there is no request, the node passes the token to its successor as usual. If there is one request, the token holder sets its successor node to be the requesting node and transmits the token to it; the requestor sets its linkages accordingly and proceeds. If more than one node requests to enter the ring, the token holder will detect a garbled transmission. The conflict is resolved by an address-based contention scheme: The token holder transmits a *resolve-contention* packet and waits four response windows. Each requestor can respond in one of these windows, based on the first two bits of its address. If a requestor hears anything before its window comes up, it refrains from requesting entrance. If the token holder receives a valid response, it is in business; otherwise it tries again, and only those nodes that requested the first time are allowed to request this time, based on the second pair of bits in their address. This process continues until a valid request is received, no request is received, or a maximum retry count is reached. In the latter two cases, the token holder gives up and passes the token to its logical successor in the ring.

Deletion from ring is much simpler. If a node wishes to drop out, it waits until it receives the token, then sends a *set-successor* packet to its predecessor, instructing it to splice to its successor.

Fault management by the token holder covers a number of contingencies. First, while holding the token, a node may hear a packet, indicating that another node has the token. In this case, it immediately drops the token by reverting to listener mode, and the number of token holders drops immediately to 1 or 0. Upon completion of its turn, the token holder will issue a token packet to its successor. The successor should immediately issue a data or token packet. Therefore, after sending a token, the token issuer will listen for one slot time to make sure that its successor is active. This precipitates a sequence of events:

(1) If the successor node is active, the token issuer will hear a valid packet and revert to listener mode.
(2) If the issuer does not hear a valid packet, it reissues the token to the same successor one more time.
(3) After two failures, the issuer assumes that its successor has failed and issues a *who-follows* packet, asking for the identity of the node that follows the failed node. The issuer should get back a set-successor packet from the second node down the line. If so, the issuer adjusts its linkage and issues a token (back to Step 1).
(4) If the issuing node gets no response to its who-follows packets, it tries again.
(5) If the who-follows tactic fails, the node issues a solicit-successor packet with the full address range (i.e., every node is invited to respond). If this process works, a two-node ring is established and the procedure continues.
(6) If two attempts of Step 5 fail, the node assumes that a catastrophe has occurred; perhaps the node's receiver has failed. In any case, the node ceases activity and listens to the bus.

Logical *ring initialization* occurs when one or more stations detect a lack of bus activity lasting longer than a time-out value: The token has been lost. This can

Table 6. Ring Access Methods

Characteristic	Register insertion	Slotted ring	Token ring
Transmit opportunity	Idle state plus empty buffer	Empty slot	Delimiter with free indicator
Frame purge responsibility	Receiver or transmitter	Transmitter node	Transmitter node
Principal advantage	Maximum ring utilization	Simplicity	Regulated but fair access
Principal disadvantage	Purge mechanism	Bandwidth waste	Token monitor required

result from a number of causes, for example, the network has just been powered up, or a token-holding station fails. Once its time-out expires, a node will issue a *claim-token* packet. Contending claimants are resolved in a manner similar to the response-window process.

4.1.3 CSMA/CD versus Token Bus

At present, CSMA/CD and token bus are the two principal contenders for medium access control technique on bus/tree topologies. In this section we examine their comparative disadvantages and advantages.

It should be obvious that the principal disadvantage of token bus is its complexity; the logic at each station far exceeds that required for CSMA/CD. A second disadvantage is the overhead involved; under lightly loaded conditions, a station may have to wait through many fruitless token passes for a turn.

Indeed, in considering these disadvantages, it would seem difficult to make a case for token bus. Such a case can be made, however [Miller and Thompson 1982; Stieglitz 1981], and includes the following elements. First, it is easy to regulate the traffic in a token bus system; different stations can be allowed to hold the token for differing amounts of time. Second, unlike CSMA/CD, there is no minimum packet length requirement with token bus. Third, the requirement for listening while talking imposes physical and electrical constraints on the CSMA/CD system that do not apply to token systems. Finally, token bus is significantly superior to CSMA/CD under heavy loads, as discussed below.

Another advertised advantage of token bus is that it is "deterministic"; that is, there is a known upper bound to the amount of time that any station must wait before transmitting. This upper bound is known because each station in the logical ring only can hold the token for a specified time. In contrast, the delay time with CSMA/CD can only be expressed statistically, and since every attempt to transmit under CSMA/CD can, in principle, produce a collision, there is a possibility that a station could be shut out indefinitely. For process control and other real-time applications, this "nondeterministic" behavior is undesirable. Unfortunately, in actual application there is always a finite possibility of transmission error, which can cause a lost token. This adds a statistical component to the behavior of a token bus system.

4.2 Ring LANs

Over the years, a number of different algorithms have been proposed for controlling access to the ring (surveys are presented by Penny and Baghdadi [1979] and Liu [1978]). The three most common ring access techniques are discussed in this section: token ring, register insertion, and slotted ring. In Table 6 these three methods are compared on a number of characteristics:

Transmit Opportunity. When may a repeater insert a packet onto the ring?

Packet Purge Responsibility. Who removes a packet from a ring to avoid its circulating indefinitely?

Number of Packets on Ring. This depends not only on the "bit length" of the ring relative to the packet length, but on the access method.

Principal Advantage.

Principal Disadvantage.

The significance of the table entries will become clear as the discussion proceeds.

4.2.1 Token Ring

The token ring is probably the oldest ring control technique, originally proposed in 1969 [Farmer and Newhall 1969] and referred to as the Newhall ring. This has become the most popular ring access technique in the United States. Prime Computer [Gordon et al. 1980] and Apollo both market token ring products, and IBM seems committed to such a product [Rauch-Hinden 1982]. This technique is the ring access method selected for standardization by the IEEE 802 Local Network Standards Committee [Andrews and Shultz 1982; Dixon 1982; Markov and Strole 1982].

The token ring technique is based on the use of a small token packet that circulates around the ring: When all stations are idle, the token packet is labeled as a "free" token. A station wishing to transmit waits until it detects the token passing by, alters the bit pattern of the token from "free token" to "busy token," and transmits a packet immediately following the busy token (Figure 14).

There is now no free token on the ring, and so other stations wishing to transmit must wait. The packet on the ring will make a round trip and be purged by the transmitting station. The transmitting station will insert a new free token on the ring when both of the following conditions have been met:

- the station has completed transmission of its packet, and
- the busy token has returned to the station.

If the bit length of the ring is less than the packet length, then the first condition implies the second. If not, then a station could release a free token after it has finished transmitting but before it receives its own busy token. However, this might complicate error recovery, since several packets will be on the ring at the same time. In any case, the use of a token guarantees that only one station at a time may transmit.

When a transmitting station releases a new free token, the next station downstream with data to send will be able to seize the token and transmit.

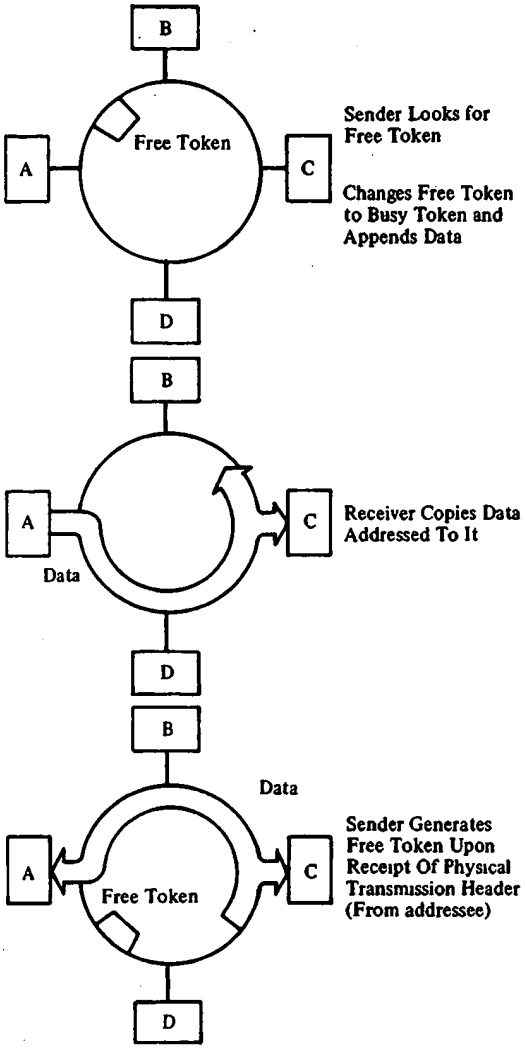


Figure 14. Token ring.

One nice feature of token ring, and all ring protocols, is this: If the source station has responsibility for removing the circulating packet, then the destination station can set bits in the packet as it goes by to inform the sender of the result of the transmission. For example, the IEEE 802 standard includes three such control bits in the packet format: address recognized (A), packet copied (C), and error (E). The A and C bits allow the source station to differentiate three conditions:

- station nonexistent/nonactive,
- station exists but packet not copied,
- packet received.

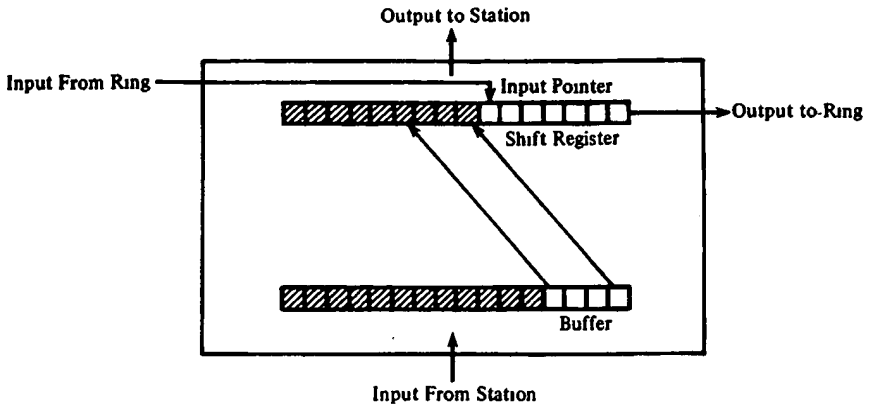


Figure 15. Register insertion technique.

The E bit can be set by any station if an error is detected.

There are two error conditions that could cause the token ring system to break down. One is the loss of the token so that no token is circulating; the other is a persistent busy token that circulates endlessly. To overcome these problems the IEEE 802 standard specifies that one station be designated as "active monitor." The monitor detects the lost token condition using a time-out, and recovers by issuing a new free token. To detect a circulating busy token, the monitor sets a "monitor bit" to one on any passing busy token. If it sees a busy token with the bit already set, it knows that the transmitting station has failed to absorb its packet and recovers by changing the busy token to a free token.

Other stations on the ring have the role of passive monitor. Their primary job is to detect failure of the active monitor and assume that role. A contention-resolution algorithm is used to determine which station takes over.

The token ring technique shares many of the advantages of token bus. Perhaps its principal advantage is that traffic can be regulated, either by allowing stations to transmit differing numbers of data when they receive the token, or by setting priorities so that higher priority stations have first claim on a circulating token.

The principal disadvantage of token ring is the requirement for token maintenance. The active/passive monitor system described above must be employed for error recovery.

4.2.2 *Register Insertion*

The register insertion strategy was originally developed by researchers at Ohio State University [Hafner et al. 1974], and is the technique used in the IBM Series 1 product [IBM 1982] and a Swiss product called SILK [Huber et al. 1983]. It derives its name from the shift register associated with each station on the ring, which is equal in size to the maximum packet length, and used for temporarily holding packets that circulate past the station. In addition, the station has a buffer for storing locally produced packets.

The register insertion ring can be explained with reference to Figure 15, which shows the shift register and buffer at one station. First, consider the case in which the station has no data to send, but is merely handling packets of data that circulate by its position. When the ring is idle, the input pointer points to the rightmost position of the shift register, indicating that it is empty. When a packet arrives along the ring, it is inserted bit by bit in the shift register, with the input pointer shifting left for each bit. The packet begins with an address field. As soon as the entire address field is in the buffer, the station can determine if it is the addressee. If not, then the

packet is forwarded by shifting one bit out on the right as each new bit arrives from the left, with the input pointer stationary. After the last bit of the packet has arrived, the station continues to shift bits out to the right until the packet is gone. If no additional packets arrive during this time, the input pointer will return to its initial position. Otherwise, a second packet will begin to accumulate in the register as the first is shifted out.

If the arriving packet is addressed to the node in question, it can either erase the address bits from the shift register and divert the remainder of the packet to itself, thus purging the packet from the ring, or retransmit the data as before, while copying it to the local station.

Now consider the case in which the station has data to transmit. A packet to be transmitted is placed in the output buffer. If the line is idle and the shift register is empty, the packet can be transferred immediately to the shift register. If the packet consists of some length n bits, less than the maximum frame size, and if at least n bits are empty in the shift register, the n bits are parallel transferred to the empty portion of the shift register immediately adjacent to the full portion; the input pointer is adjusted accordingly.

The principal advantage of the register insertion technique is that it achieves the maximum ring utilization of any of the methods. A station may transmit whenever the ring is idle at its location, and multiple packets may be on the ring at any one time.

The principal disadvantage is the purge mechanism. Allowing multiple packets on the ring requires the recognition of an address prior to removal of a packet. If a packet's address field is damaged, it could circulate indefinitely. One possible solution is the use of an error-detecting code on the address field.

4.2.3 *Slotted Ring*

The slotted ring technique was first developed by Pierce [1972], and is sometimes referred to as the Pierce loop. Most of the development work on this technique was done at the University of Cambridge in England [Hopper 1977; Wilkes and Wheeler 1979], and a number of British firms market commercial versions of the Cambridge ring [Heywood 1981].

In the slotted ring, a number of fixed-length slots circulate continuously on the ring. Each slot contains a leading bit to designate the slot as empty or full. All slots are initially marked empty. A station wishing to transmit waits until an empty slot arrives, marks the slot full, and inserts a packet of data as the slot goes by. The station cannot transmit another packet until this slot returns. The slot may also contain response bits, which can be set on the fly by the addressed station to indicate accepted, busy, or rejected. The full slot makes a complete round trip, to be marked empty again by the source. Each station knows the total number of slots on the ring and can thus clear the full/empty bit that it had set as it goes by. Once the now empty slot goes by, the station is free to transmit again.

In the Cambridge ring, each slot contains room for one source and one destination address byte, two data bytes, and five control bits for a total length of 37 bits.

The Cambridge ring contains several interesting features. A station may decide that it wishes to receive data only from one other station. To accomplish this each station includes a source select register. When this register contains all ones, the station will receive a packet addressed to it from any source; when it contains all zeros, the station will not accept packets from any source. Otherwise the station is open to receive packets only from the source whose address is specified by the register.

In addition, the Cambridge ring specifies two response bits in each packet to differentiate four conditions:

- destination nonexistent/nonactive,
- packet accepted,
- destination exists but packet not accepted,
- destination busy.

Finally, the Cambridge ring includes a monitor, whose task it is to empty a slot that is persistently full.

The principal disadvantage of the slotted ring is that it is wasteful of bandwidth.

First, a slot typically contains more overhead bits than data bits. Second, a station may send only one packet per round-trip ring time. If only one or a few stations have packets to transmit, many of the slots will circulate empty.

The principal advantage of the slotted ring appears to be its simplicity. The interaction with the ring at each node is minimized, improving reliability.

4.3 HSLNs

In this section, we review the only technique that thus far has gained favor for HSLNs, known as prioritized CSMA. It is also referred to as CSMA with collision avoidance. The technique will be described in terms of the ANSI draft standard [Burr 1983]; the algorithm for HYPERchannel is very similar.

The protocol is based on CSMA; that is, a station wishing to transmit listens to the medium and defers if a transmission is in progress. In addition, an algorithm is used that specifically seeks to avoid collisions when the medium is found idle by multiple stations.

For this scheme, the stations or ports form an ordered logical sequence (PORT(1), PORT(2), . . . , PORT(N)), which need not correspond to physical position of the bus.

The scheme is initialized after each transmission by any port. Following initialization, each station, in turn, may transmit if none of the stations in sequence before it have done so. So PORT(I + 1) waits until after PORT(I) has had a chance to transmit. The waiting time consists of

(1) the earliest time at which PORT(I) could begin transmitting (which depends on the transmission opportunity for PORT(I − 1)), plus
(2) a port delay time during which PORT(I) has the opportunity to transmit, plus
(3) the propagation delay between the two ports.

As we shall see, this rather simple concept becomes more complex as we consider its refinements.

The basic rule can be described as follows. After any transmission, PORT(1) has the right to transmit. If it fails to do so in a reasonable time, then PORT(2) has the chance, and so on. If any port transmits, the system reinitializes.

The first refinement is to permit multiframe dialogues. To accommodate this, an additional rule is added: After any transmission, the port receiving that transmission has the first right to transmit. If that port fails to transmit, then it is the turn of PORT(1), and so on. This will permit two ports to seize the medium, with one port sending data frames and the other sending acknowledgment frames.

The second refinement is to accommodate the case in which nobody has a frame to transmit. HYPERchannel solves this by entering a free-for-all period, in which collisions are allowed. ANSI has a more elegant solution: If none of the stations transmit when they have an opportunity, then the highest priority station times out and transmits a dummy frame. This serves to reinitialize the network and start over.

With these two refinements, we can depict the MAC protocol as a simple sequence of events:

1. Medium is active
2. Medium Goes Idle
   If Receiver Transmits,
   Then Go To 1
   Else Go To 3
3. If PORT(1) Transmits,
   Then Go To 1
   Else Go To 4

   ⋮

N + 3. If No Port Transmits, Then Go To 1.

The third refinement has to do with balancing this scheme, which at this point is biased to the lower number ports [PORT(1) *always* gets a shot, for example]. To make the scheme fair, a port that has just transmitted should not try again until everyone else has had a chance.

To concisely describe the algorithm with these three refinements, we need to define some quantities:

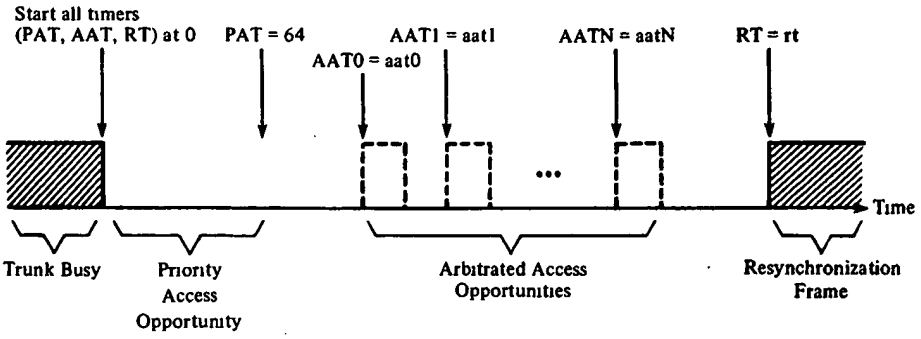


Figure 16. Operation of the ANSI medium access control protocol.

Priority Access Opportunity. A period of time granted to a port after it receives a frame. May be used to acknowledge frame and/or continue a multiframe dialogue.

Priority Access Timer (PAT). Used to time priority access opportunities: 64-bit times.

Arbitrated Access Opportunity. A period of time granted to each port in sequence, during which it may initate a transmission: 16-bit times. Assigned to individual ports to avoid collisions.

Arbitrated Access Timer (AAT). Used to provide each port with a unique, nonoverlapping, arbitrated access opportunity.

Resynchronization Timer (RT). Time by which the latest possible arbitrated transmission should have been received. Used to reset all timers.

Arbiter Wait Flag (WF). Used to enforce fairness. When a port transmits, its WF is set so that it will not attempt another arbitrated transmission until all other ports have an opportunity.

Figure 16 depicts the process of this refined algorithm. For the timers listed, we use the convention that uppercase letters refer to the variable name and lowercase letters to a specific value. A timer that reaches a specified maximum value is said to have expired.

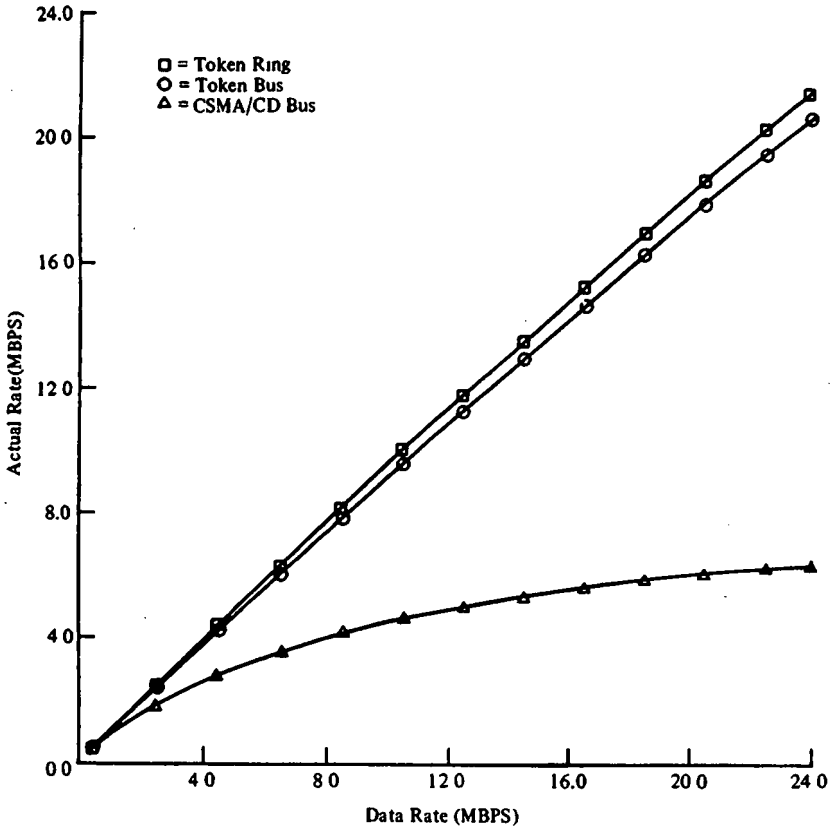
This technique seems well suited to HSLN requirements. The provision for multiframe dialogue permits rapid transfer of large files. A typical HSLN consists of only a small number of stations; under these circumstances, the round-robin access technique will not result in undue delays.

5. COMPARATIVE PERFORMANCE OF LAN PROTOCOLS

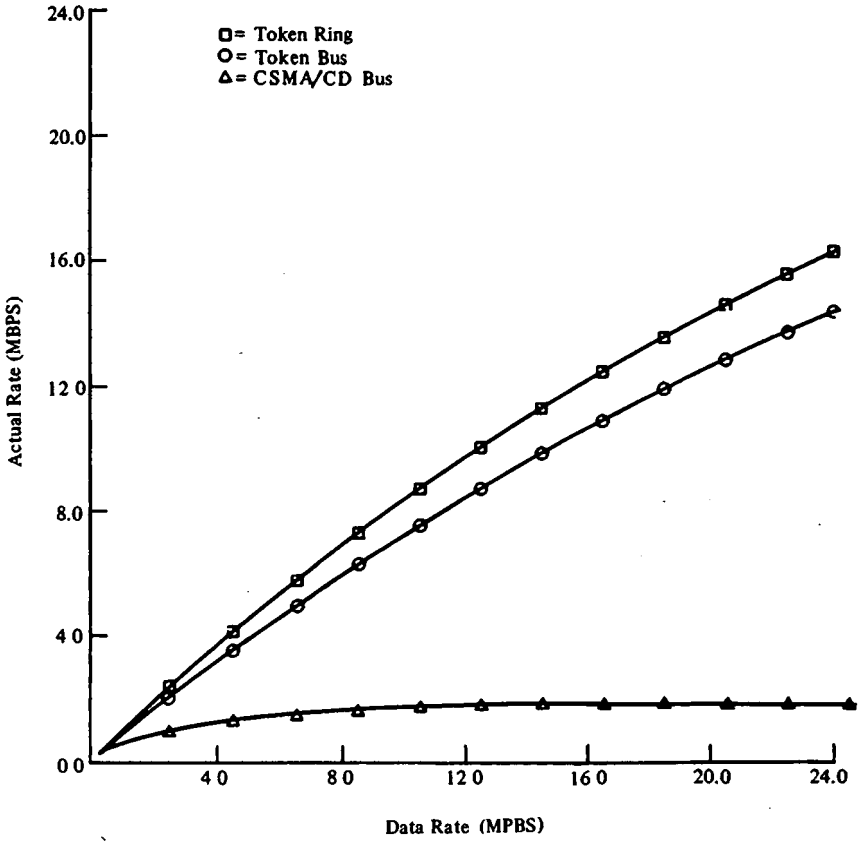
Although there have been a number of performance studies focusing on a single protocol, there have been few systematic attempts to analyze the relative performance of the various local network protocols. In what follows, we look at the results of several carefully done studies that have produced comparative results. A lengthy survey of performance studies is found in Tropper [1981].

5.1 CSMA/CD, Token Bus, and Token Ring

One important study was done by a group at Bell Laboratories, under the sponsorship of the IEEE 802 Local Network Standards Committee [Arthurs and Stuck 1981; Stuck 1983a, 1983b]. Naturally enough, the study analyzed the three protocols being standardized by IEEE 802: CSMA/CD, token bus, and token ring. Two cases of message arrival statistics are employed. In the first case, only one out of one hundred stations has data to transmit, and is always ready to transmit. In such a case, one would hope that the network would not be the bottleneck, but could easily keep up with one station. In the second case, all one hundred stations always have data to transmit. This represents an extreme of congestion, and one would expect that the network may be a bottleneck. In both cases, the one station or one hundred stations provide enough


□ = Token Ring
○ = Token Bus
△ = CSMA/CD Bus
Actual Rate(MBPS)
24.0
20 0
16 0
12 0
8.0
4 0
0 0
4 0
8 0
12 0
16.0
20 0
24 0
Data Rate (MBPS)

(a)


□= Token Ring
○= Token Bus
△= CSMA/CD Bus
Actual Rate (MBPS)
24.0
20.0
16.0
12 0
8.0
4 0
0 0
4 0
8 0
12 0
16 0
20.0
24.0
Data Rate (MPBS)

(b)

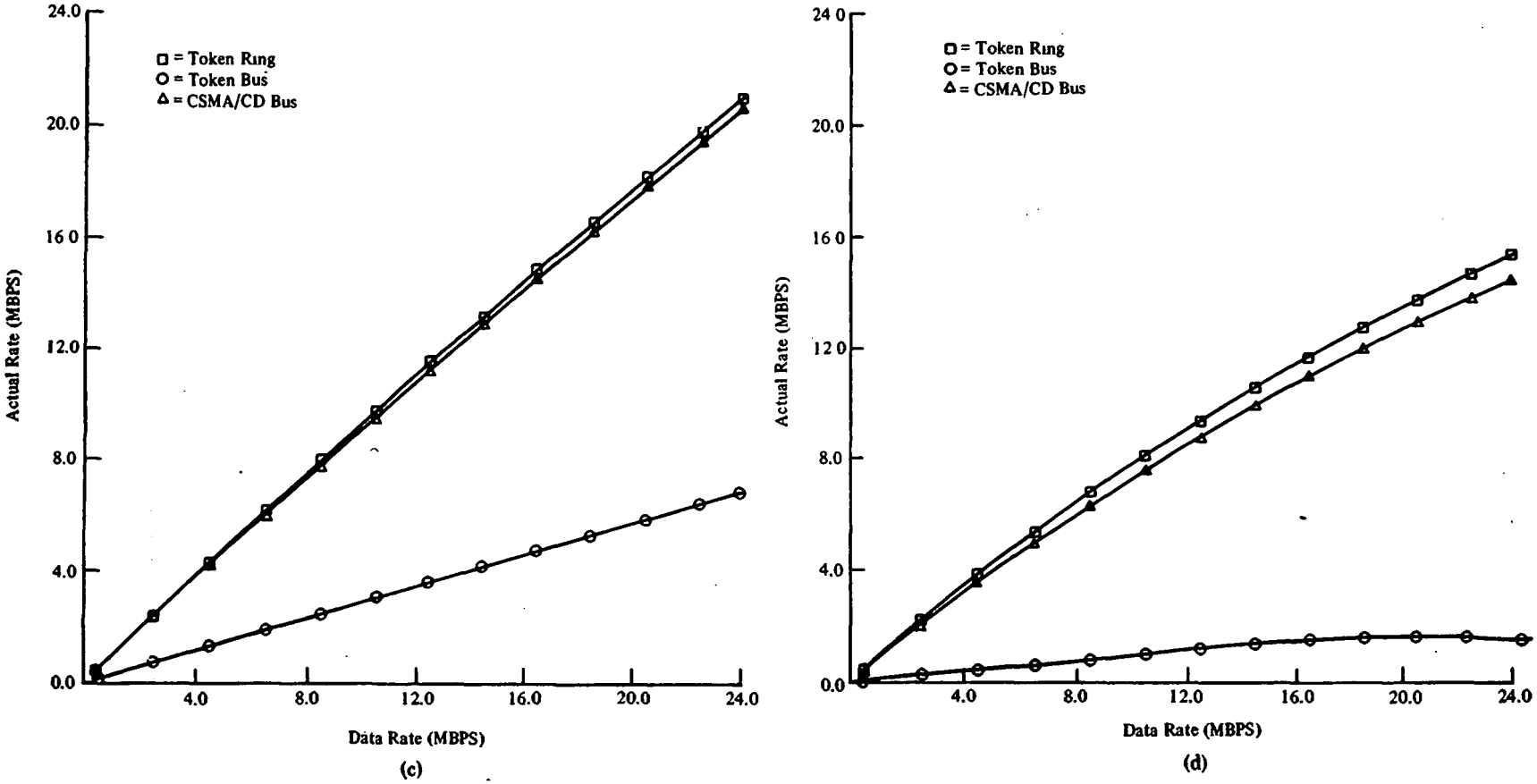


Figure 17. Potential throughput of IEEE 802 protocols: (a) 2000 bits per packet, 100 stations active out of 100 stations total; (b) 500 bits per packet, 100 stations active out of 100 stations total; (c) 2000 bits per packet, 1 station active out of 100 stations total; (d) 500 bits per packet, 1 station active out of 100 stations total.

input to fully utilize the network. Hence the results are a measure of maximum potential utilization. In addition, results were obtained by using two different packet sizes to determine the relative effect of this parameter on the three protocols. Thus a total of four different cases was analyzed.

The results are given in Figure 17, which shows the actual data transmission rate versus the transmission speed of the medium for the four cases. The length of the medium (bus or ring) is assumed to be 2 kilometers. Note that the abscissa in the plots is not offered load but the actual capacity of the medium. Three systems are examined: token ring with a 1-bit latency per station, token bus, and CSMA/CD. The analysis yields the following conclusions:

(1) For the given parameters, the smaller the mean packet length, the greater is the difference in maximum mean throughput rate between token passing and CSMA/CD. This reflects the fact that, for a given amount of data, smaller packets mean more packets and hence more collisions under CSMA/CD.
(2) Token ring is the least sensitive to workload.
(3) CSMA/CD offers the shortest delay under light load, whereas it is most sensitive under heavy load to the workload.

Note also that in the case of a single station transmitting, token bus is significantly less efficient than the other two protocols. This is so because the assumption is made that the delay in token processing is greater for token bus than that for token ring.

Another interesting phenomenon is seen most clearly in Figure 17b. For a CSMA/CD system under these conditions, the maximum effective throughput at 5 Mbps is only about 1.25 Mbps. If expected load is, say, 0.75 Mbps, this configuration may be perfectly adequate. If however, the load is expected to grow to 2 Mbps, raising the network data rate to 10 Mbps or even 20 Mbps will not accommodate the increase!

The reason for this disparity between CSMA/CD and token passing (bus or ring) under heavy load has to do with the instability of CSMA/CD. As offered load increases, so does throughput, until, beyond some maximum value, throughput actually declines as offered load increases. This results from the fact that there is an increased frequency of collisions: More packets are offered, but fewer successfully escape collision. Worse, those packets that do collide must be retransmitted, further increasing the load.

5.2 CSMA/CD and Ring Protocols

It is far more difficult to do a comparative performance of the three major ring protocols than to do a comparison of bus and token ring protocols, as the results depend on a number of parameters unique to each protocol, for example,

- *Token Ring.* Size of token, token processing time.
- *Slotted Ring.* Slot size, overhead bits per slot.
- *Register Insertion.* Register size.

Although there have been a number of studies on each one of the techniques, few have attempted pairwise comparisons, much less a three-way analysis. The most systematic work in this area has been done by Liu and his associates [Liu et al. 1982]. Liu made comparisons based on analytic models developed by others for token ring, slotted ring, and CSMA/CD, plus his own formulations for register insertion. He then obtained very good corroboration from simulation studies.

Figure 18 is a summary of the results, on the basis of the assumption that relatively large packets are used and that register insertion ring packets are removed by the destination station, whereas slotted ring and token ring packets are removed by the source station. This is clearly an unfair comparison since register insertion under this scheme does not include acknowledgments, whereas token ring and slotted ring do. The figure does show that slotted ring is the poorest performer, and that register insertion can carry a load greater than 1.0, since the protocol permits multiple packets to circulate.

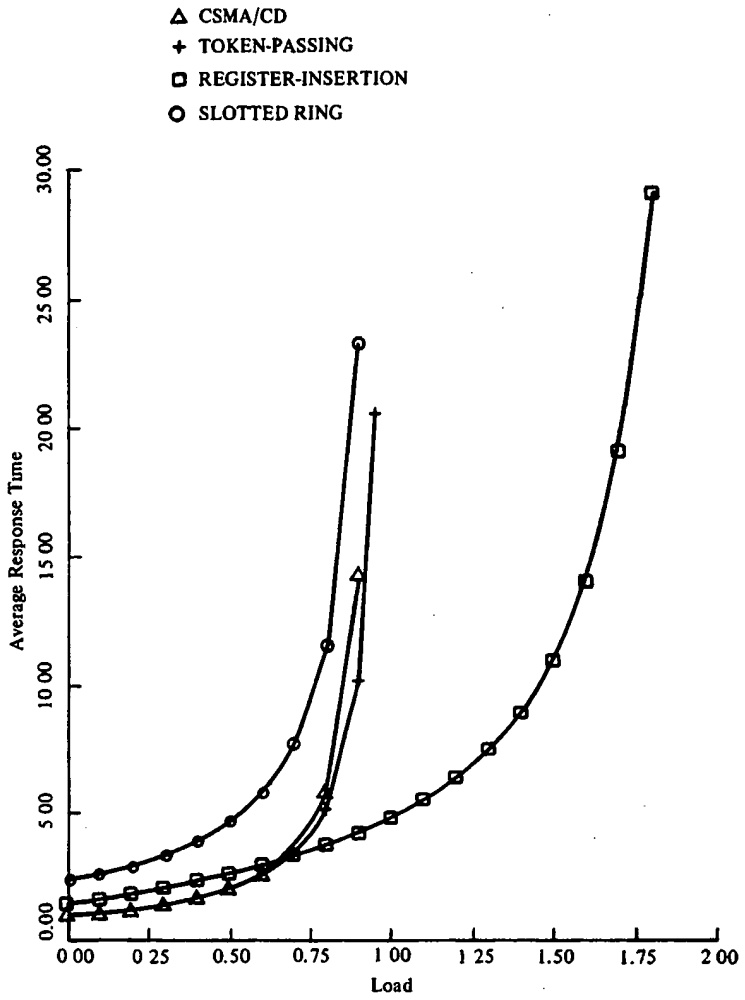


Figure 18. Delay for various protocols.

Bux performed an analysis comparing token ring, slotted ring, and CSMA/CD, yielding results similar to those of Liu [Bux 1981]. He confirmed several important conclusions: that token ring suffers greater delay than CSMA/CD at light load but less delay and more stable throughput at heavy loads, and further, that token ring has superior delay characteristics to slotted ring.

It is difficult to draw conclusions from the efforts made thus far. The slotted ring seems to be the least desirable over a broad range of parameter values, owing to the considerable overhead associated with each small packet. As between token ring and register insertion, the evidence suggests that at least for some sets of parameter values, register insertion gives superior throughput and delay performance.

6. STANDARDS

6.1 Local Area Networks

The key to the development of the LAN market is the availability of a low-cost interface; the cost to connect equipment to a LAN must be much less than the cost of the actual equipment. This requirement, as

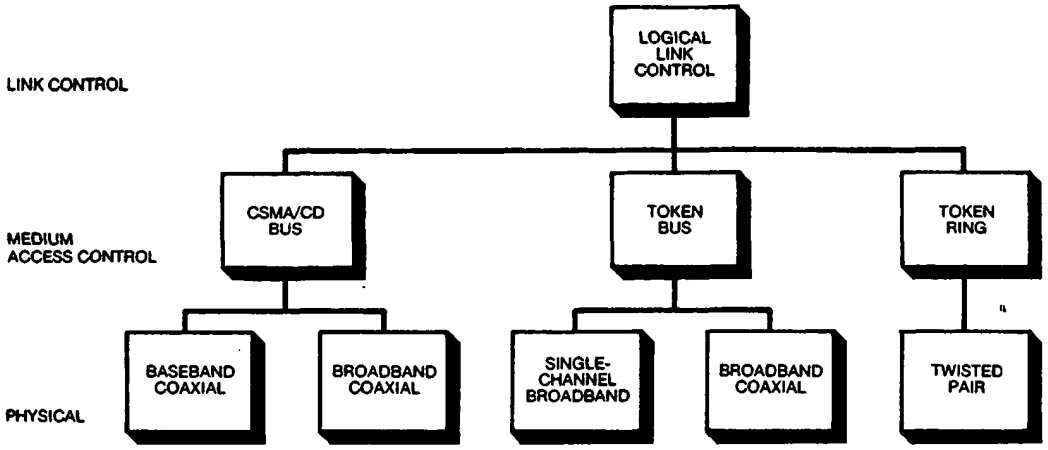


Figure 19. The IEEE 802 standard.

well as the complexity of the LAN protocols, dictates a very large scale integration (VLSI) solution. However, chip manufacturers are reluctant to commit the necessary resources unless there is a high-volume market. A LAN standard would ensure that volume and also enable equipment of a variety of manufacturers to intercommunicate.

This is the rationale of the IEEE 802 committee [Clancy et al. 1982], which has developed a set of standards for LANs [IEEE 1983]. The standards are in the form of a three-layer communications architecture with a treelike expansion of options from top to bottom (Figure 19). The three layers, known as logical link control, medium access control, and physical, encompass the functionality of the lowest two layers (data link and physical) of the International Organization for Standardization (ISO) reference model.

The logical link control (LLC) layer provides for the exchange of data between service access points (SAPs), which are multiplexed over a single physical connection to the LAN. The LLC provides for both a connectionless, datagramlike service, and a connection-oriented, virtual-circuit-like service. Both the protocol and the frame format resemble HDLC.

At the medium access control layer, there are three standards: CSMA/CD has converged with the Ethernet specification, which is well suited to typical office applications, and two forms of token access have been standardized, for time-critical applications, such as process control, as well as for office applications.

For CSMA/CD, a 10-Mbps baseband physical layer has been approved. Several broadband options are still under consideration, ranging in data rate per channel from 1 to 5 Mbps.

For token bus, three physical layers are provided as options. The simplest and least expensive is a single-channel broadband system using frequency shift keying (FSK) at 1 Mbps. A more expensive version of this system runs at 5 or 10 Mbps and is intended to be easily upgradable to the final option, which is multichannel broadband. The latter provides data rates of 1, 5, or 10 Mbps.

For token ring, twisted pair has been defined, which will provide data rates of 1 and 4 Mbps.

The range of options offered may be disheartening to the reader. However, the IEEE 802 Committee has at least narrowed the alternatives, and it is to be expected that the bulk of future LAN development work will be within the scope laid down by IEEE 802.

More detailed discussions of the standard may be found in Nelson [1983], Graube [1982], Allan [1982], and Myers [1982].

6.2 High-Speed Local Networks

For HSLNs, the necessity for standards seems less compelling. Because of the high data rate requirement, the HSLN vendor must provide a high-throughput interface to the attached device. Such an interface has a cost in the tens of thousands of dollars range, and a VLSI protocol implementation will not significantly affect the price.

The X.3T9.5 Committee sponsored by ANSI has prepared a draft HSLN standard [ANSI 1982; Burr 1983; Parker 1983] using a two-layer model, corresponding to the data link and physical layers of the ISO model. The data link layer specifies a simple connectionless service. The physical layer specification includes the collision-avoidance protocol discussed earlier, and it specifies a data rate of 50 Mbps.

7. SUMMARY

Local networks can be characterized, in large measure, by the transmission medium and topology employed. Common combinations are

Twisted Pair Bus or Ring. An inexpensive, easily installed network for a small number of low-cost, low-throughput devices.

Baseband Coaxial Cable Bus or Ring. A general-purpose network; supports moderate numbers of devices over moderate distances; suitable for many office applications.

High-Speed Baseband Coaxial Cable Bus. Supports a small number of high-throughput devices; suitable for computer-room requirements.

Broadband Coaxial Cable Bus. The most flexible and general-purpose network; supports a large number of devices over a wide area, and can handle a variety of traffic requirements using FDM.

Twisted Pair Star. The architecture of the CBX. Suitable for supporting at moderate cost a large number of limited throughput devices.

For descriptive clarity, and to reflect differences in standards and technology, three categories of local networks can be defined. The category local area network (LAN) covers a variety of local networks using a bus, tree, or ring topology and that support a variety of applications and loads. The high-speed local network (HSLN) is specifically designed to support high-speed data transfer among a limited number of devices. The third category, the computerized branch exchange (CBX), is a digital private branch exchange designed to handle both voice and data connections.

An important design issue for LANs and HSLNs is the choice of medium access control protocol. The most important ones for the bus/tree topology are CSMA/CD, token bus, and collision avoidance, and, for the ring topology, token ring, register insertion, and slotted ring.

The literature on local networks is growing as rapidly as the field itself. Annotated bibliographies may be found in Shoch [1980] and Stallings [1983]. Stallings [1984] is a textbook on the subject.

REFERENCES

ALLAN, R. 1982. Local-area networks spur moves to standardize data communications among computers and peripherals. *Electron. Des.* Dec. 23, 107–112.

ALLAN, R. 1983. Local networks: Fiber optics gains momentum. *Electron. Des.* June 23.

ANDREWS, D. W., AND SCHULTZ, G. D. 1982. A token-ring architecture for local area networks: An update. In *Proceedings of COMPCON Fall 82* (Washington, D.C., Sept. 20–23). IEEE Computer Society, Los Angeles, pp. 615–624.

ANSI 1982. *Draft Proposed American National Standard Local Distributed Data Interface.* American National Standards Institute, New York.

ARTHURS, E., AND STUCK, B. W. 1981. A theoretical performance analysis of polling and carrier sense collision detection communication systems. In *Proceedings of the 7th Symposium on Data Communications* (Mexico City, Oct. 27–29). *Comput. Commun. Rev. 11*, 4, 156–163.

BOSEN, R. 1981. A low-speed local net for under $100 per station. *Data Commun. 10*, 2 (Dec.), 81–83.

BURR, W. 1983. An overview of the proposed american national standard for local distributed data interfaces. *Commun. ACM, 26*, 10 (Oct.), 554–561.

BUX, W. 1981. Local-area subnetworks: A performance comparison. *IEEE Trans. Commun. COM-29*, 10 (Oct.).

BUX, W., CLOSS, F., JANSON, P. A., KUMMERLE, K., MILLER, H. R., AND ROTHAUSER, H. 1982. A

local-area communication network based on a reliable token ring system. In *Proceedings of the International Symposium on Local Computer Networks.*

CHRISTENSEN, G. S. 1979. Links between computer-room networks. *Telecommunications 13*, 2 (Feb.), 47–50.

CLANCY, G. J., et al. 1982. The IEEE 802 Committee states its case concerning its local network standards efforts. *Data Commun. 11*, 4 (Apr.), 13, 238.

COOPER, E. 1982. 13 often-asked questions about broadband. *Data Commun 11*, 4 (Apr.), 137–142.

COOPER, E. 1983. Broadband network design: Issues and answers. *Comput. Des. 22*, 3 (Mar.), 209–216.

COOPER, E., AND EDHOLM, P. 1983. Design issues in broadband local networks. *Data Commun. 12*, 2 (Feb.), 109–122.

DEC 1980. *The Ethernet: A Local Area Network Data Link Layer and Physical Layer Specifications*, Sept. 30. Digital Equipment Corp., Intil Corp., and Xerox Corp., Digital Equipment Corp., Maynard, Mass.

DERFLER, F., AND STALLINGS, W. 1983. *A Manager's Guide to Local Networks.* Prentice-Hall, New York.

DINESON, M. A., AND PICAZO, J. J. 1980. Broadband technology magnifies local network capability. *Data Commun. 9*, 2 (Feb.), 61–79.

DIXON, R. C. 1982. Ring network topology for local data communications. In *Proceedings of COMPCON, Fall 82* (Washington, D.C., Sept. 20–23). IEEE Computer Society, Los Angeles, pp. 591–605.

FARMER, W. D., AND NEWHALL, E. E. 1969. An experimental distributed switching system to handle bursty computer traffic. In *Proceedings of the ACM Symposium on Problems in the Optimization of Data Communications.* ACM, New York.

FORBES, J. 1981. RF prescribed for many local links. *Data Commun. 10*, 9 (Sept.).

FRANTA, W. R., AND CHLAMTEC, I. 1981. *Local Networks.* Lexington Books, Lexington, Mass.

FREEDMAN, D. 1983. Fiber optics shine in local area networks. *Mini-Micro Syst. 16*, 10 (Sept.), 225–230.

GORDON, R. L., FARR, W. W., AND LEVINE, P. 1980. Ringnet: A packet switched local network with decentralized control. *Comput. Networks 3*, 373–379.

GRAUBE, M. 1982. Local area nets: A pair of standards. *IEEE Spectrum* (June), 60–64.

HAFNER, E. R., NENADAL, Z., AND TSCHANZ, M. 1974. A digital loop communications system. *IEEE Trans. Commun. COM-22*, 6 (June), 877–881.

HAHN, M., AND BELANGER, P. 1981. Network minimizes overhead of small computers. *Electronics*, Aug. 25.

HEYWOOD, P. 1981. The Cambridge ring is still making the rounds. *Data Commun. 10*, 7 (July), 32–36.

HOHN, W. C. 1980. The Control Data loosely coupled network lower level protocols. In *Proceedings of the National Computer Conference* (Anaheim, Calif., May 19–22), vol. 49. AFIPS Press, Reston, Va., 129–134.

HOPKINS, G. T. 1979. Multimode communications on the MITRENET. *Proceedings of the Local Area Communications Network Symposium* (Boston, May). Mitre Corp., McLean, Va., pp. 169–178.

HOPKINS, G. T., AND MEISNER, N. B. 1982. Choosing between broadband and baseband local networks. *Mini-Micro Syst. 16*, 7 (June).

HOPPER, A. 1977. Data ring at Computer Laboratory, University of Cambridge. In *Local Area Networking.* NBS Publ. National Bureau of Standards, Washington, D.C., pp. 500–531, 11–16.

HUBER, D., STEINLIN, W., AND WILD, P. 1983. SILK: An implementation of a buffer insertion ring. *IEEE J. Selected Areas Commun. SAC-1*, 5 (Nov.), 766–774.

IBM CORP. 1982. *IBM Series/1 Local Communications Controller Feature Description.* GA34-0142-2. IBM Corporation.

IEEE 1983. *IEEE Project 802, Local Network Standards.* Institute of Electrical and Electronic Engineers, New York.

KRUTSCH, T. E. 1981. A user speaks out: Broadband or baseband for local nets? *Data Commun. 10*, 12 (Dec.), 105–112.

LIU, M. T. 1978. Distributed loop computer networks. In *Advances in Computers*, vol. 17. Academic Press, New York, pp. 163–221.

LIU, M. T., HILAL, W., AND GROOMES, B. H. 1982. Performance evaluation of channel access protocols for local computer networks. In *Proceedings of COMPCON FALL 82* (Washington, D.C., Sept. 20–23). IEEE Computer Society, Los Angeles, pp. 417–426.

LUCZAK, E. C. 1978. Global bus computer communication techniques. In *Proceedings of the Symposium on Computer Networks* (Gaithersburg, Md., Dec.). IEEE Computer Society, Los Angeles, pp. 58–67.

MALONE, J. 1981. The microcomputer connection to local networks. *Data Commun. 10*, 12 (Dec.), 101–104.

MARKOV, J. D., AND STROLE, N. C. 1982. Token-ring local area networks: A perspective. In *Proceedings of COMPCON FALL 82* (Washington, D.C., Sept. 20–23). IEEE Computer Society, Los Angeles, pp. 606–614.

METCALFE, R. M., AND BOGGS, D. R. 1976. Ethernet: Distributed packet switching for local computer networks. *Commun. ACM 19*, 7 (July), 395–404.

METCALFE, R. M., BOGGS, D. R., THACKER, C. P., AND LAMPSON, B. W. 1977. Multipoint data communication system with collision detection. U.S. Patent 4,063,220.

MILLER, C. K., AND THOMPSON, D. M. 1982. Making a case for token passing in local networks. *Data Commun. 11*, 3 (Mar.), 79–88.

MYERS, W. 1982. Towards a local network standard. *IEEE Micro 2*, 3 (Aug.), 28–45.

NELSON, J. 1983. 802: A progress report. *Datamation* (Sept.), 136–152.

PARKER, R. 1983. Committees push to standardize disk I/O. *Comput Des 22*, 3 (Mar.), 30–34.

PENNY, B. K., AND BAGHDADI, A. A. 1979. Survey of computer communications loop networks. *Comput. Commun. 2*, 4 (Aug.), 165–180; *2*, 4 (Oct.), 224–241.

PIERCE, J. R. 1972. Network for block switches of data. *Bell Syst. Tech. J 51*, 6 (July–Aug.).

RAUCH-HINDIN, W. 1982. IBM's local network scheme. *Data Commun. 11*, 5 (May), 65–70.

RAWSON, E., AND METCALFE, R. 1978. Fibernet: Multimode optical fibers for local computer networks. *IEEE Trans. Commun. COM-26*, 7 (July), 983–990.

ROMAN, G. S. 1977. *The design of broadband coaxial cable networks for multimode communications.* MITRE Tech. Rep. MTR-3527. Mitre Corp., McLean, Va.

ROSENTHAL, R., Ed. 1982. The selection of local area computer networks. NBS Special Publ. 500-96, National Bureau of Standards, Washington, D.C., Nov.

SALTZER, J. H., AND CLARK, D. D. 1981. Why a ring? In *Proceedings of the 7th Symposium on Data Communications* (Mexico City, Oct. 27–29). *Comput. Commun. Rev. 11*, 4, 211–217.

SALTZER, J. H., AND POGRAN, K. T. 1979. A star-shaped ring network with high maintainability. In *Proceedings of the Symposium on Local Area Communications Network* (Boston, May). Mitre Corp., McLean, Va., pp. 179–190.

SALWEN, H. 1983. In praise of ring architecture for local area networks. *Comput. Des. 22*, 3 (Mar.), 183–192.

SHOCH, J. F. 1980. *An Annotated Bibliography on Local Computer Networks.* Xerox Palo Alto Research Center, Palo Alto, Calif., Apr.

SHOCH, J. F., DALA, Y. K., AND REDELL, D. D. 1982. Evolution of the Ethernet local computer network. *Computer 15*, 8 (Aug.), pp. 1–27.

STAHLMAN, M. 1982. Inside Wang's local net architecture. *Data Commun. 11*, 1 (Jan.), 85–90.

STALLINGS, W. 1983. *Tutorial: Local Network Technology.* IEEE Computer Society Press, Silver Spring, Md.

STALLINGS, W. 1984. *Local Networks: An Introduction* Macmillan, New York.

STIEGLITZ, M. 1981. Local network access tradeoffs. *Comput. Des 20*, 12 (Oct.), 163–168.

STUCK, B. 1983a. Which local net bus access is most sensitive to congestion? *Data Commun. 12*, 1 (Jan.), 107–120.

STUCK, B. 1983b. Calculating the maximum mean data rate in local area networks. *Computer 16*, 5 (May), 72–76.

THORNTON, J. E. 1980. Back-end network approaches. *Computer 13*, 2 (Feb.), 10–17.

TROPPER, C. 1981. *Local Computer Network Technologies* Academic Press, New York.

WILKES, M. V., AND WHEELER, D. J. 1979. The Cambridge digital communication ring. In *Proceedings of the Symposium on Local Area Communications Network* (Boston, May). Mitre Corp., McLean, Va., pp. 47–62.

YEN, C., AND CRAWFORD, R. 1983. Distribution and equalization of signal on coaxial cables used in 10 Mbits baseband local area networks. *IEEE Trans. Commun. COM-31*, 10 (Oct.), 1181–1186.

Received January 1983; final revision accepted March 1984